UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Procesamiento del Lenguaje Natural para la extracción en tiempo real de neologismos del español en redes sociales

Daniel Santaella Santos
Tutor: Pablo Alfonso Haya Coll
Ponente: Germán Montoro Manrique

Enero 2019

Procesamiento del Lenguaje Natural para la extracción en tiempo real de neologismos del español en redes sociales

AUTOR: Daniel Santaella Santos
TUTOR: Pablo Alfonso Haya Coll

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Enero de 2019

Resumen (Castellano)

Este Trabajo de Fin de Grado (TFG) y su temática giran en torno al desarrollo, mejora, y despliegue de una herramienta informática accesible en línea que nos permitirá ahondar en profundidad en el descubrimiento de términos de nuevo uso del castellano, también conocidos como neologismos. Éstos no solo incluyen el descubrimiento de palabras de nuevo uso, sino también la inclusión de términos de otras lenguas como pueden ser los anglicismos o de nuevas acepciones para palabras ya existentes, surgidas de una constante evolución lingüística por parte de todos los hispanohablantes del mundo.

Para que la herramienta web tuviera la orientación adecuada desde el primer momento, era necesario recoger la mayor cantidad de datos posibles del mayor número de hablantes posibles. Las redes sociales son un punto de partida inigualable debido a su gran variedad lingüística. Estas plataformas permiten recoger el uso que las personas hacen de todos los tipos de términos, ya sean inventados o no, dándole nuevas acepciones o modificando las ya existentes, así como abrazando términos de otros idiomas y adaptándolos al nuestro. En nuestro caso, se decidió que Twitter sería la red social predefinida para analizar la estructura y el contexto de las palabras.

El análisis del idioma nos lleva a descubrir muchas peculiaridades y rasgos del grupo de personas que lo habla: el lenguaje nos descubre elementos comunes entre humanos para comunicar experiencias y vivencias. El hecho de descubrir neologismos implica, en cierto modo, cambio, evolución. Sin duda, es difícil descubrir cuándo el giro en una de estas palabras implica novedad en el lenguaje. Es aquí cuando entran en escena los expertos en la materia, que se apoyarán en la herramienta para poder facilitar su tarea de descubrimiento de nuevos términos y su catalogación, todo ello basado en tecnologías en auge utilizadas en multitud de sectores tecnológicos en la actualidad.

Por lo tanto, en esta memoria se recogerán todos los aspectos a destacar en la creación de la herramienta LECONORE (LÉxico COloquial NO REgistrado), incluyendo las tecnologías y lenguajes utilizados en la primera línea de código hasta la propia experiencia de uso de la aplicación desarrollada en este TFG.

Palabras clave (Castellano)

Neologismo, anglicismo, redes sociales, Twitter, adopción, sinonimia, acepción.

Abstract (English)

This Final Project and its theme revolve around the development, improvement and deployment of a computer tool available online that will allow us to research deeply into the discovery of terms of new use in Spanish, also known as neologisms. These not only include the discovery of words of new use, but also the inclusion of terms of other languages such as English terms or new meanings for existing words, arising from a constant linguistic evolution by all Spanish speakers all around the world.

In order for the web tool to have the right orientation from the beginning, it was necessary to collect as much data as possible from as many speakers as possible. Social networks are an unbeatable starting point due to its great linguistic variety. These platforms allow collecting the use that people make of all types of terms, whether invented or not, giving new meanings or modifying existing ones, as well as embracing terms of other languages and adapting them to ours. In our case, it was decided that Twitter would be the predefined social network to analyze the structure and context of the words.

The analysis of the language leads us to discover many peculiarities and features of the group of people who speak it: language reveals common elements amongst humans to communicate experiences and thoughts. The discovery of neologisms implies, in a certain way, change, evolution. Undoubtedly, it is difficult to discover when the meaning-twist in one of these words implies novelty in the language. This is when the experts in the field come into scene, who will rely on the tool to facilitate their task of discovering new terms and their cataloging, all based on state-of-art technologies used in many technological sectors of the economy today.

Therefore, in this report we will gather all aspects to be highlighted in the creation of the tool knows as LECONORE (Unregistered Colloquial Lexicon in Spanish), including the technologies and coding languages used in the very first line of code to the user experience when using the application developed in this End-Of-Grade Project.

Keywords (English)

Neologism, anglicism, social networks, Twitter, adoption, synonymy, meaning.

Agradecimientos

Me gustaría, en primer lugar, agradecer a mi familia la paciencia que ha tenido a lo largo de todos estos años y el apoyo incondicional que ha sido para mí, en especial a mi madre.

Mencionar a todas aquellas personas con las que he compartido experiencias y vivencias desde que entramos en a la Escuela Politécnica Superior, desde las buenas hasta las malas, y siguen siendo parte de mi, aun habiendo acabado los estudios y tomado cada uno su propio camino. Sabéis quiénes sois.

Aprovecho también para agradecer a mi tutor, Pablo A. Haya el desempeño impecable en su labor docente, así como al resto de profesores de los que realmente he aprendido y me han ayudado a crecer.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

Esta memoria de TFG hace referencia al trabajo de desarrollo e implementación de LECONORE, acrónimo de LÉxico COloquial NO REgistrado, una herramienta web que ayudará al descubrimiento de palabras de nuevo uso en el castellano, de nuevas acepciones de palabras ya existentes, así como extranjerismos adoptados por nuestro idioma. Es importante, sin embargo, que el descubrimiento o incorporación de estas palabras respondan a unos criterios comunes, ordenados, y que refleje fielmente las nuevas necesidades expresivas de los hablantes acomodándose al máximo a los rasgos que caracterizan al español.

Por ello, el desarrollo de la herramienta LECONORE está estrechamente ligado a la búsqueda de una fuente de información que provea un flujo constante de información a analizar: la escritura a través de redes sociales está llena de expresiones, coloquialismos, jergas y formas de hablar que se asemejan, en bastantes ocasiones, a conversaciones habladas. Si además valoramos que cualquier persona, de cualquier nivel educativo, puede expresarse libremente a través de estas redes, hemos encontrado la fuente de información pública de discurso coloquial actualizada más grande hoy en día.

Twitter ha sido la red social escogida no solo debido a su gran número de usuarios hispanohablantes, donde el español es el segundo idioma más usado, sino también por disponer de una API sencilla que facilita su integración con nuestra herramienta de descubrimiento de neologismos. [1]

Por todo ello, expertos en la materia del *Laboratorio de Lingüística Informática* de la Universidad Autónoma de Madrid en colaboración con el grupo Esvarés de la Universidad de Salamanca, tratarán de beneficiarse de la tecnología al servicio de la lengua.

1.2 Objetivos

El propósito de las tareas a realizar como parte del Trabajo de Fin de Grado es continuar con el desarrollo de funcionalidades de LECONORE, comenzadas en un TFG anterior '*Herramienta de extracción de neologismos del español en redes sociales*'. Se han definido varias líneas de actuación:

- **Rediseño de la interfaz gráfica** centrada en la experiencia del usuario: una interfaz intuitiva, amigable y adaptada para distintos dispositivos, que muestre información y estadísticas de una manera clara y concisa, facilitando el uso y comprensión por parte de usuarios no técnicos.

- **Monitorización en tiempo real**, lo cual implica el desarrollo de un método de conexión a Twitter para obtener tweets y analizarlos, básico para el funcionamiento del programa.

- **Revisión de funcionalidades del PLN** (Procesador de Lenguaje Natural) para abrir el abanico de candidatos que son detectados tras el análisis de tweets.

- **Mantenimiento y corrección** de errores que se descubren a medida que se hace un mayor uso de la aplicación.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos, que a su vez deja entrever las distintas fases por las que ha pasado el desarrollo de la herramienta LECONORE:

- **Capítulo 1: Introducción**

  En este capítulo se recoge la información básica que facilita la comprensión del desarrollo y estudio que se ha realizado sobre este TFG y los objetivos a llevar a cabo.

- **Capítulo 2: Estado del Arte**

  Esta fase de documentación recoge un resumen de las nuevas tecnologías, herramientas y lenguajes actuales en los que se basa este proyecto.

- **Capítulo 3: Análisis**

  En este apartado se incluye el análisis de los requisitos, el alcance del proyecto, así como los requisitos funcionales y no funcionales. Se realiza un estudio de la estructura inicial de proyecto con la finalidad de establecer los pasos a seguir a la hora de implementar las mejoras propuestas.

- **Capítulo 4: Diseño**

  Se hará mención del diseño de los componentes informáticos del sistema que dan respuesta a las funcionalidades descritas en los capítulos anteriores detallando en la medida de lo posible la relación y formas de interacción entre todas ellas.

- **Capítulo 5: Desarrollo**

  Dicho capítulo abarca los procedimientos utilizados y decisiones tomadas para el cumplimiento de los objetivos propuestos en los diferentes subsistemas mediante un seguimiento continuo de los objetivos. Asimismo, se mencionan aspectos relevantes de la implementación de ciertos módulos de vital importancia para la aplicación.

- **Capítulo 6: Integración, pruebas y resultados**

  Se exponen las diferentes pruebas realizadas durante el desarrollo del proyecto para asegurar la eficiencia, funcionamiento y resultados obtenidos de los módulos extendidos, así como de la aplicación en su conjunto.

- **Capítulo 7: Conclusiones, discusión y trabajo futuro**

  Finalmente, en la última sección se hará una breve síntesis de los resultados finales del presente proyecto, así como mencionar las diferentes posibilidades de mejora, ampliación de funcionalidades y cualquier detalle relevante sobre todo el proceso realizado.

2 Estado del arte

En esta sección comienza con una breve introducción a conceptos lingüísticos y técnicos utilizados en el resto del documento, seguido de un planteamiento de los problemas y objetivos. Se menta, además, las propuestas a nivel tecnológico para solventarlos.

2.1 Definiciones y conceptos básicos

A continuación, se detallan algunos conceptos genéricos relacionados con el ámbito de este documento:

Definición 2.1.1 – Según la RAE, un neologismo es un vocablo, acepción o giro nuevo en una lengua.

Definición 2.1.2 – JSON, acrónimo de *JavaScript Object Notation*, es un formato de texto ligero para el intercambio de datos.

Definición 2.1.3 – PLN, o Procesamiento Lenguaje Natural, es un módulo ya existente en el programa que se encarga de ingerir y procesar la información exterior de varias maneras, ya sea mediante la aplicación LECONORE o mediante un script de carga de datos. El propósito de este motor, que recibe *tuits* en formato JSON, es pasarlos por una serie de filtros que decidirán si una palabra es candidata a ser un neologismo o, por el contrario, no puede ser considerada como tal.

Definición 2.1.4 – Microblogging, en español microblogueo, es el concepto de publicar o enviar mensajes breves.

Definición 2.1.5 – API, o *Application Programming Interface*, es el conjunto de subrutinas, procedimiento y funciones que ofrece cierta biblioteca para que distintos programas informáticos puedan comunicarse entre ellos, siguiendo estas reglas.

Definición 2.1.6 – Se llama *framework* o entorno de trabajo a un conjunto estandarizado de concepto y prácticas que sirven como esqueleto y patrón para el desarrollo e implementación de una aplicación informática.

2.2 Twitter

Esta red social, catalogada como servicio de microblogging, dispone de diversos términos y nomenclaturas que conviene explicar, de cara a mejorar el entendimiento global sobre la finalidad de nuestra aplicación, que se nutre de ella.

2.2.1 Términos de Twitter

A continuación, se detalla en qué consisten algunos términos propios del uso de esta red social, así como del funcionamiento de la API con la cuál se han obtenido los tuits a analizar.

Definición 2.2.1.1 – Un **tweet**, o *tuit*, es un mensaje corto 280 caracteres como máximo (originalmente 140) que un usuario comparte en su página principal en Twitter que puede contener fotos, videos, links o GIFs.

Definición 2.2.1.2 – **Seguidor**, o seguidor, son aquellas personas que te siguen o siguen a alguien.

Definición 2.2.1.3 – **Following**, o seguidos, son aquellas personas a las que sigues.

Definición 2.2.1.4 – **Time Line** (TL), en español *Línea de Tiempo*, también conocido coloquialmente como muro, es el espacio donde van apareciendo las actualizaciones de la gente a la que sigues.

Definición 2.2.1.5 – **Trending Topics** (TT), en español *Tópicos Tendencia*, son las palabras o frases más utilizadas en el momento por la comunidad de usuarios de Twitter. Es un buen punto de partida para encontrar temas actuales.

Definición 2.2.1.6 – **Hashtags** (#), son los símbolos que se colocan delante de palabras para convertirlas en una etiqueta de tipo enlace.

2.2.2 API de Twitter

Para poder interactuar con Twitter de una manera menos convencional y más amplia, las empresas, desarrolladores y usuarios finales disponen de una serie de herramientas que les facilitan esta tarea. Estas APIs de Twitter permiten acceder a diversas funcionalidades para crear un programa que se integre con la red social del pájaro azul.

Existen varias APIs [1], así como planes Standard y Premium de estas, en función de las tareas a desarrollar:

- ***Search Tweets*** permite buscar tuits históricos en la última semana, el último mes o desde el mismo 2006.

- ***Filter realtime Tweets*** selecciona únicamente los tuits que te interesan mediante el uso de filtros avanzados como palabras clave, localizaciones o idioma con la API de streaming en tiempo real.

- ***Account Activity API*** para obtener la actividad de una cuenta, ***Direct Message API*** para obtener experiencias de usuario personalizadas, así como interacciones más innovadoras, ***Twitter for websites*** para embeber Tweets, Time Lines y mucho más en cualquier página web o ***Ads API*** para la programación, creación y gestión de campañas de anuncios en la red social.

Asimismo, existen también bibliotecas en muchos lenguajes de programación que facilitan el uso de estas APIs. En nuestro caso, en consonancia con la recomendación del tutor, hemos utilizado Tweepy [1], ya que no hemos encontrado ninguna funcionalidad importante que nos haya llevado a hacer uso de otras APIs como Twython o Python-Twitter.

A la hora de querer realizar cualquier acción mediante estas bibliotecas, en primer lugar, se ha de disponer de una cuenta de Twitter y habilitarla para hacer uso de desarrollador. De esta

manera, y tras rellenar unos formularios informando a Twitter del uso que vas a hacer de sus datos, se te proporcionarán unos *tokens* o códigos imprescindibles para la comunicación con la API.

Una vez autenticados podremos hacer una búsqueda mediante la API indicando obligatoriamente una *query* o palabra clave. Es posible agregar multitud de filtros opcionales en función de nuestras necesidades: restringir la búsqueda a tuits en un idioma en concreto, limitar el número de tuits a devolver por búsqueda, devolver los tuits de usuarios en una zona concreta a partir de una longitud, latitud y radio determinada, etc.

Podemos encontrar un ejemplo real de la respuesta a una petición para obtener un tweet con la API de Tweepy en formato JSON en el *apéndice A*.

2.3 Elasticsearch y Kibana

Elasticsearch es un potente motor de búsqueda de código abierto que nos permite indexar y recuperar grandes cantidades de información en formato JSON. Esta tecnología se basa en Apache Lucene, para proveer a diferentes aplicaciones de un motor de búsqueda flexible, fiable y rápido. Posee, además, multitud de características que lo hacen ventajoso y adecuado para encajarlo en nuestro proyecto, entre las que destacar:

- Arquitectura escalable, distribuida y en alta disponibilidad, lo que nos garantiza la existencia del dato.

- Capacidad analítica y de búsqueda en tiempo real.

- API RESTfull sobre HTTP que permite integrar Elasticsearch con otras tecnologías.

- Capacidad de indexación de un documento JSON independientemente del formato de sus campos al ser considerado un repositorio no rígido de información. Además, este formato evitaría la necesidad de lanzar consultas complejas sobre una base de datos.

- Permite uno o varios índices en un clúster, así como la búsqueda sobre uno o varios índices.

- Utilidad en aplicaciones NoSQL, ya que no requiere definir y crear tablas o estructuras especificas antes de empezar a trabajar en el proyecto. Por lo tanto, cualquier modificación a posteriori en la estructura de los datos no implicaría un cambio y adaptación completas que, en caso de disponer de cantidades enormes de datos, sería un procedimiento largo y costoso.

Por otro lado, *Kibana* también es una herramienta de código abierto perteneciente a Elastic Stack, que facilita la visualización de información almacenada en tiempo real en Elasticsearch, a modo de add-on. Haciendo uso de este software, podremos hacer una primera exploración de los datos: seleccionar cualquier índice, filtrar por campos específicos, buscar en un campo específico, un texto o varios en varios campos, incluso completar con wildcards, o valores desconocidos.

Disponemos también de un panel de administración que permite modificar propiedades de datos importados, crear datos, visualizarlos en forma de diferentes gráficos de barras, de tarta, tablas, mapas o redes, etc. Al tener toda esta cantidad de información a nuestro alcance, podremos configurar un tablero o *dashboard* que nos proveerá con información valiosa para nuestra entrada constante de datos.

2.3.1 Conceptos de Elasticsearch y Kibana

Se explican una serie de conceptos [1] fundamentales de cara a entender mejor en qué consiste el proyecto y facilitar el aprendizaje de las tecnologías que se han utilizado.

Definición 2.3.1.1 – Un ***índice***, de manera simplificada, es el equivalente a una tabla en el modelo de bases de datos relacionales, es decir, un mecanismo que permite organizar los datos de una manera en concreto. Asimismo, es un espacio de nombre lógico que se asigna a uno o más fragmentos primarios y puede tener cero o más fragmentos réplica.

Definición 2.3.1.2 – Un ***clúster*** consiste en uno o más nodos que comparten el mismo nombre de clúster. Apuntan a un nodo maestro que es reemplazado en caso de fallo por otro nodo.

Definición 2.3.1.2 – Un ***nodo*** es un solo servidor o instancia en ejecución de Elasticsearch que pertenece a un clúster. Al inicio, un nodo utilizará unicast para descubrir si existe un clúster con el mismo nombre de clúster para tratar de unirse.

Definición 2.3.1.2 – Un ***fragmento*** o *shard* es una instancia única de Lucene. Está catalogada como una unidad de trabajo de bajo nivel administrada de manera automática por Elasticsearch, es decir, a la hora de realizar operaciones con datos no hay que hacerlas directamente contra un fragmento ya que estos son gestionados de manera transparente. Elasticsearch distribuye fragmentos entre todos los nodos de un clúster, siendo posible incluso moverlos o redistribuirlos entre nodos en caso de fallo de alguno de ellos.

Definición 2.3.1.2 – Un ***documento*** es un documento JSON que se almacena en Elasticsearch, como si fuera una fila en una tabla de una base de datos relacional, y contiene cero o más campos. Cada documento está almacenado en un índice y tiene un id y un tipo.

Definición 2.3.1.2 – Cada documento es almacenado en un ***fragmento primario*** o *primary shard*. Cuando se indexa un documento, primero se indexa en el fragmento primario y posteriormente en sus réplicas. Por defecto, un índice tiene cinco fragmentos primarios, número que podemos modificar para escalar el número de documentos que el índice puede manejar. Una vez creado el índice, este número no se puede modificar.

Definición 2.3.1.2 – Cada fragmento primario puede tener cero o más ***fragmentos réplica*** o *replica shards*. Una réplica es una copia de uno primario, y tiene dos finalidades: aumentar la conmutación por error o tolerancia en caso de que un primario falle, y aumentar el rendimiento y la cantidad de operaciones al poder cambiar dinámicamente el número de réplicas, a diferencia de los primarios.

2.4 Tecnologías para el desarrollo web

2.4.1 Python

Es un lenguaje de programación interpretado y multiplataforma creado a finales de los ochenta por Guido Van Rossum, gran aficionado del grupo cómico inglés *Monty Python*. Soporta la orientación a objetos, aunque también puede ser utilizado para programación funcional. Es de código abierto y utiliza una licencia compatible con la licencia GNU pública. Este lenguaje posee una serie de reglas de estilo para escribir código fuente de una manera ordenada, clara, sencilla y legible [1].

Una de las características del código escrito en Python es su legibilidad y transparencia, que facilita el entendimiento a alto nivel. Además, gestiona la memoria utilizando tipado dinámico y conteo de referencias. Es un lenguaje fácil de extender, permitiendo crear módulos en otros lenguajes como C o C++.

Posee un entorno de desarrollo integrado o IDLE (Integrated DeveLopment Environment) incorporado dentro de la propia instalación de Python que permite escribir y ejecutar código o cargar módulos complejos para poder probarlos de una manera rápida y sencilla.

2.4.2 Django

Django es un framework para el desarrollo web escrito en Python, interpretado, multiplataforma y orientado a objetos. De código abierto, sigue el patrón Model-Template-View (MTV), muy similar a Model-View-Controller (MVC) de Java.

Este patrón MTV [1] es muy típico y se puede ver en mucho de los frameworks de desarrollo modernos. De forma más detallada:

- **Model** o Modelo: hace referencia a los datos que maneja nuestra aplicación web y cómo se almacenan. Es la capa de abstracción encargada de la interacción y comunicación con la base de datos que utilizamos en la aplicación, en este caso Elasticsearch, pero por defecto utilizaría BBDD en lenguaje SQL.

- **Template** o Plantilla: se encarga de gestionar la representación de los datos en nuestra web. Las tecnologías implicadas en este desarrollo en particular han sido HTML5, CSS y Bootstrap, pero no se limita únicamente a estas. Podría haberse implementado alguna funcionalidad en JavaScript o Ajax, para la actualización de elementos concretos de nuestra página sin recargarla completamente.

- **View** o Vistas: en este módulo se ubica la lógica de nuestra aplicación, a través del cual se intercambiará la información de nuestros modelos a nuestras plantillas.

Como podemos observar en la *figura 4-1*, el usuario hace uso del navegador para acceder a cierta URL. Esta dirección llama a una vista encargada de la lógica de la aplicación que, a su vez, a través del modelo, obtiene la información requerida de la base de datos en uso. La API escrita en Python para Elasticsearch se encargará de devolver la información necesaria

en unos contextos que son invocados desde una plantilla, encargada de mostrar la información con el formato de visualización adecuado.

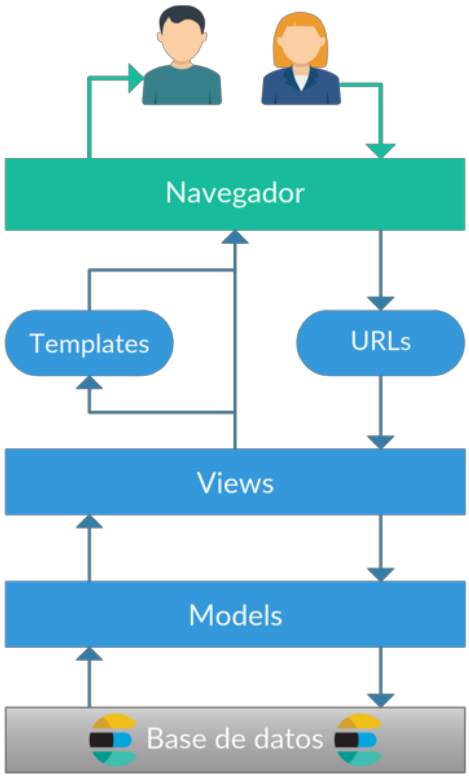


Figura 2-1: Modelo MTV

Este framework funciona sobre distintas versiones de Python, desde la 2.7 a la 3. En nuestro caso hemos realizado la instalación de este software mediante la herramienta de gestión de paquetes para Python ***pip***, con el comando ***pip3 install Django***.

Si no se tiene instalada la herramienta *pip* (la cual viene de serie a partir de Python 3.4) siempre podemos descargar el script mediante el comando con permisos de administrador ***python get-pip.py***.

La comprobación más sencilla para saber si la instalación de Django se ha ejecutado correctamente ejecutar el siguiente comando para ver la versión de instalación.

$ python3 -c "import django; print(django.get_version())"

2.4.3 HTML5

El HTML5 o *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto) es la quinta versión del lenguaje de programación básico para páginas web. Es usado para representar y estructurar el contenido en forma de texto, así como para complementarlo con objetos como pueden ser imágenes. Surgió en el 1991 y, desde entonces, su desarrollo ha pasado por varias fases hasta que en el 2008 salió la versión 5, utilizada en la implementación

de este proyecto, con nuevos elementos y características sintácticas que han permitido crear sitios modernos que mejoren la experiencia de un usuario de internet.

Su nueva API permite la adición de funcionalidades como áreas de dibujo o canvas, servicios de edición de documentos, tecnologías de almacenamiento en la nube y de mensajería web, elementos multimedia como video y audio, formularios mejorados para definir mejor los elementos que ser capturan de la interfaz y validación directa, y todo ello sin la necesidad de instalar software propietario o instalar algún plug-in.

Entre otras mejoras que nos han llevado a utilizar HTML5 en nuestra página, aparte de su integración con el framework de Django, son los menores tiempos de carga, una mejor estructura de las páginas con etiquetas específicas y el uso de imágenes dinámicas.

2.4.4 CSS

Las siglas de CSS se corresponden con la expresión ingles *Cascading StyleSheets*, u Hojas de Estilo en Cascada. Este concepto es usado en informática para referirse a un lenguaje de diseño gráfico para representar documentos escritos en un lenguaje de marcado como puede ser HTML5.

CSS fue creada por el W3C (World Wide Web Consortium) para permitir la separación de los contenidos de los documentos de la propia presentación del documento. Podremos definir, por lo tanto, elementos tales como bordes, márgenes, fondos, colores, tipo de letra, bordes y sus tamaños, permitiéndole a los desarrolladores un control más sencillo del estilo y forma de sus páginas o documentos.

Sin entrar en mayor detalle, cabe mencionar que el lenguaje se basa en una serie de reglas consistentes en un *selector* y una *declaración*: el primero indica los elementos HTML que van a estar afectados por la declaración, y el último define la información de estilo de cómo se verá el selector.

2.4.5 Bootstrap

Este framework de código abierto para el diseño de sitios y aplicaciones web con CSS y JavaScript ha permitido que mejoremos la página de LECONORE mediante un rediseño de la interfaz gráfica adaptándolo a los nuevos tiempos. La página es *responsive* y se visualiza correctamente desde el dispositivo en el que se visualice.

Creado inicialmente por trabajadores de Twitter en el 2011, este software permite crear diseños simples, claros e intuitivos, lo que facilita la rapidez de carga de los distintos elementos.

2.5 Docker

Esta tecnología de código abierto sirve para la creación de imágenes o *contenedores* de aplicaciones para automatizan el despliegue de soluciones software entre distintos sistemas operativos.

Estos contenedores incluyen la aplicación que se quiere hacer portable además de todos los elementos imprescindibles para que funcionen correctamente, bibliotecas incluidas: gracias a estos contenedores, es posible que desarrolladores y administradores envíen y ejecuten aplicaciones independientemente del sistema operativo.

En comparación con las máquinas virtuales, que requieren cargar y ejecutar un SO completo, los contenedores demandan muy pocos recursos, además de ser ventajosamente más rápidos en ejecutarse. En el *anexo C* se detalla cómo desplegar una imagen en cualquier máquina con el proyecto.

3 Análisis

A lo largo de este capítulo se procederá a realizar un análisis del proyecto detallando, entre otros, sus objetivos principales, su alcance, requisitos funcionales y no funcionales, casos de uso.

3.1 Objetivo del programa

La finalidad de esta aplicación, definida anteriormente por otro Trabajo de Fin de Grado y continuada mediante el presente documento, es la creación de una herramienta tecnológica que, basada en lenguajes y herramientas de desarrollo actuales y punteras, nos permita realizar una carga de tuits previamente descargados, en tiempo real para poder analizar las posibles palabras de nuevo uso, y la creación de un diccionario con los términos identificados.

3.2 Alcance

El alcance del proyecto se limita al análisis e identificación de palabras provenientes de Twitter en castellano para el descubrimiento de neologismos de nuestra lengua. Inicialmente han sido definidos dos perfiles para la aplicación: el perfil *visitante*, que permite a cualquier persona acceder a la página y navegar para visualizar las últimas palabras descubiertas, estadísticas e información de contexto sobre éstas, y el perfil *administrador*, que podrá realizar tareas más avanzadas como procesar y catalogar tuits, finalidad para la que ha sido concebida LECONORE.

3.3 Definiciones y conceptos específicos

En este apartado se hace mencionan una serie de conceptos específicos de LECONORE que ayudan a comprender más en profundidad funcionamiento de la aplicación. Es importante distinguir los matices que caracterizan cada uno de estos conceptos para entender en qué consiste esta herramienta en línea.

Definición 3.3.1 – Llamaremos *candidato* a toda aquella palabra que tras pasar el proceso de análisis del PLN sea considerada positivamente como posible neologismo.

Definición 3.3.2 – Llamaremos *descartada* a toda aquella palabra que no pasa el proceso de análisis del PLN, o bien es catalogada específicamente como tal.

Definición 3.3.3 – Llamaremos *admitida* a toda aquella palabra que tras haber pasado el proceso de análisis del PLN y ser considerada candidata, es aceptada por un lingüista o usuario administrador cualificado.

Definición 3.3.4 – En nuestra aplicación, llamaremos *neologismo* a toda aquella palabra admitida que supera un proceso de 'lematización' y pasa a contener más información acerca del propio término, como el ámbito del saber o un ejemplo de uso.

Definición 3.3.5 – Llamaremos *publicado* a todo aquel neologismo que pase a ser público en LECONORE, es decir, visible a nivel de aplicación.

3.4 Definición de requisitos

Se enumerarán tanto los requisitos funcionales como los no funcionales, basados en las reuniones y necesidades definidas entre las partes involucradas.

3.4.1 Requisitos funcionales

Responden principalmente a los requisitos de los usuarios y sus restricciones en tanto a los servicios que prestará la aplicación. Por el momento, existirán dos tipos de usuarios que pueda interactuar con la web, visitantes o administradores.

- **RF01. Visualizar estadísticas**
  Los usuarios podrán ver estadísticas acerca de los términos admitidos como candidatos, términos descartados, aceptados como neologismos, etc., independientemente de su perfil.

- **RF02. Buscador de neologismos**
  Existirá un buscador en diversas partes de la aplicación para ambos perfiles de usuario que permitirá buscar neologismos y, en caso de encontrarlos, mostrar una breve ficha informativa sobre ellos.

- **RF03. Abecedario de neologismos**
  Se mostrará para visitantes y administradores un abecedario en la aplicación con las letras coloreadas o sombreadas en función de la existencia de neologismos que empiecen por esa letra.

- **RF04. Inicio de sesión**
  Existirá una página de inicio de sesión para que los lingüistas puedan hacer log-in y diferenciarse de los visitantes, además de adquirir funcionalidad extendida.

- **RF05. Carga estática de tweets**
  Se dispondrá para el administrador de la opción de cargar ficheros con tuits a la aplicación que previamente hayan sido descargados.

- **RF06. Carga dinámica de tweets**
  Se dispondrá para el administrador de la opción de introducir al sistema tuits provenientes directamente de las APIs de la red social.

- **RF07. Catalogación de términos**
  Se permitirá al usuario administrador la catalogación de palabras en función de su conocimiento y criterio, ya sean candidatos, admitidos, descartados, neologismos o publicados.

- **RF08. Soporte con el departamento**
  Se dispondrá de una manera de contacto con el departamento mediante email.

3.4.2 Requisitos no funcionales

Este tipo de requisitos son aquellos que ayudan al cumplimiento de los requisitos funcionales e imponen restricciones en tanto que apoyen decisiones de diseño o fomenten estándares de calidad.

- **RNF1. Interfaz intuitiva**
  La web deberá tener una interfaz intuitiva en su uso: sencilla, clara, limpia y fácil de entender y usar, para cualquier tipo de usuarios.

- **RNF2. Interfaz responsiva**
  El diseño web deberá ser adaptable, es decir, la finalidad es que la página sea visualizada correctamente independientemente del dispositivo desde el que se cargue.

- **RNF3. Rendimiento**
  Debido a la gran cantidad de datos que se manejarán, es necesario crear la web de una manera que el tiempo de respuesta de ésta no limite su interactividad.

- **RNF4. Tolerancia a fallos**
  En caso de que la aplicación falle, habrá que asegurar que sea capaz de reiniciarse y recuperar la disponibilidad de esta.

- **RNF5. Seguridad**
  Únicamente los usuarios administradores deberán poder tener acceso a las funcionalidades extendidas.

- **RNF5. Escalabilidad**
  El sistema tiene que ser montado de manera que se asegure su escalabilidad, es decir, la posibilidad de añadir más usuarios registrados, o servidores de almacenamiento, sumándose a todo lo actualmente ofrecido por el sistema.

3.4.3 Requisitos de datos

Los datos para importar a la aplicación deberán respetar un formato concreto, de cara a poder ser procesados correctamente por la herramienta. El formato utilizado será JSON, ya que permite una gestión ligera y eficaz de los mismos. Hay que tener en cuenta:

- Antes de realizar un procesamiento de los tuits descargados previamente, estos ficheros pasarán a estar cargados en una carpeta local del servidor en una ruta

específica. Tras su análisis, se escribirá en el correspondiente fichero de registro para, en caso necesario, ser consultado por el administrador del servidor.

- En caso de procesar tuits descargados en tiempo real mediante la API de Twitter, éstos se volcarán a un archivo concreto en una ruta predefinida y posteriormente se analizarán de manera transparente para el usuario.

3.4.4 Rediseño interno de la aplicación

Con anterioridad a la realización de este TFG, los datos manejados en la versión previa de LECONORE se clasificaban en *candidatos*, *neologismos* y *descartados*.

Tras un análisis de los nuevos requisitos funcionales de la aplicación, se ha llegado a la conclusión de la necesidad de realizar un rediseño de la lógica interna que rige la clasificación de los distintos términos que entran a la aplicación. Estos nuevos requisitos han sido obtenidos de reuniones que han involucrado expertos en la materia del *Laboratorio de Lingüística Informática* de la Universidad Autónoma de Madrid en colaboración con el grupo Esvarés de la Universidad de Salamanca.

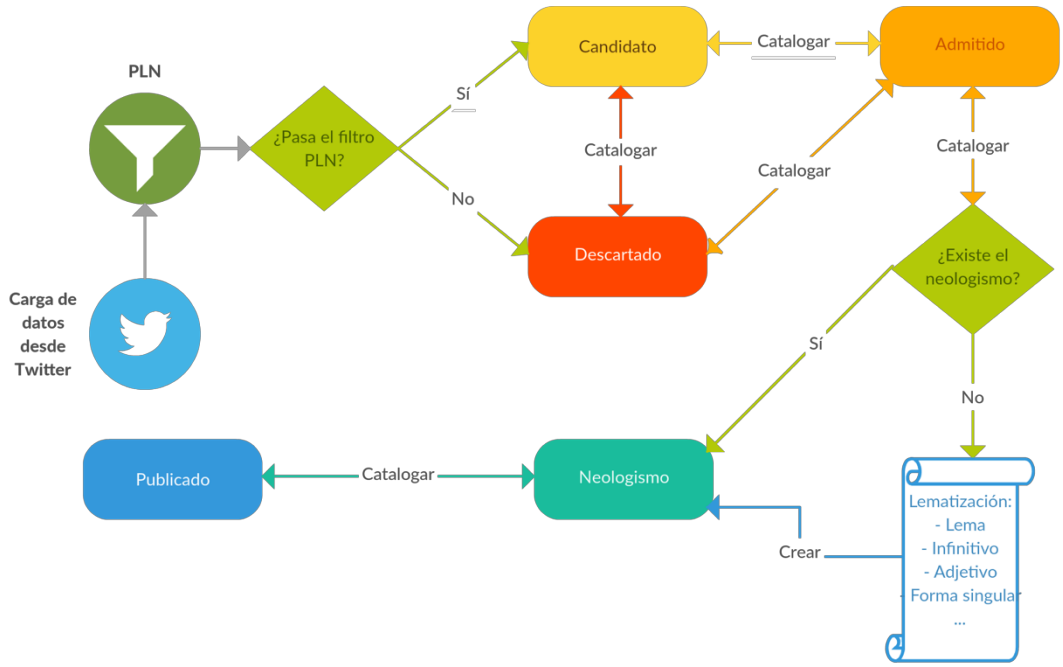


Figura 3-1: Rediseño de datos en LECONORE

Como se observa en la *figura 3-1*, ahora existen cinco estados posibles para clasificar una palabra: *candidatos*, donde inicialmente desembocan todos los términos que han superado satisfactoriamente el filtrado del PLN; *descartados* son aquellos términos que han sido declinados por el PLN, o los surgidos por rechazar un candidato; las palabras candidatas que pasan el criterio de los lingüistas son conocidas como *admitidos*; *neologismos* son aquellas palabras candidatas que sufren un proceso de lematización para proveerlas de un lema,

definición, ejemplos, categoría sintáctica y ámbito del saber, incluso variantes, observaciones morfológicas y de uso de manera opcional; por último, todos aquellos neologismos que reciben la validación final para que se muestren en las pantallas para visitantes serán conocidos como *publicados*.

3.5 Casos de uso

En esta subsección se mostrará algún caso de uso del proyecto desarrollado, mostrando las principales diferencias entre un usuario visitante y un usuario registrado. Un diagrama de caso de uso es una representación que refleja las distintas acciones que alguien tiene que llevar a cabo para realizar un proceso. Se representan, asimismo, a los diferentes actores involucrados y las acciones que realiza cada uno. En la *figura 3-2* podemos ver las distintas acciones que pueden realizar los actores de la aplicación, que incluye buscar neologismos, iniciar sesión, catalogar palabras y cargar tuits de manera estática o en vivo.

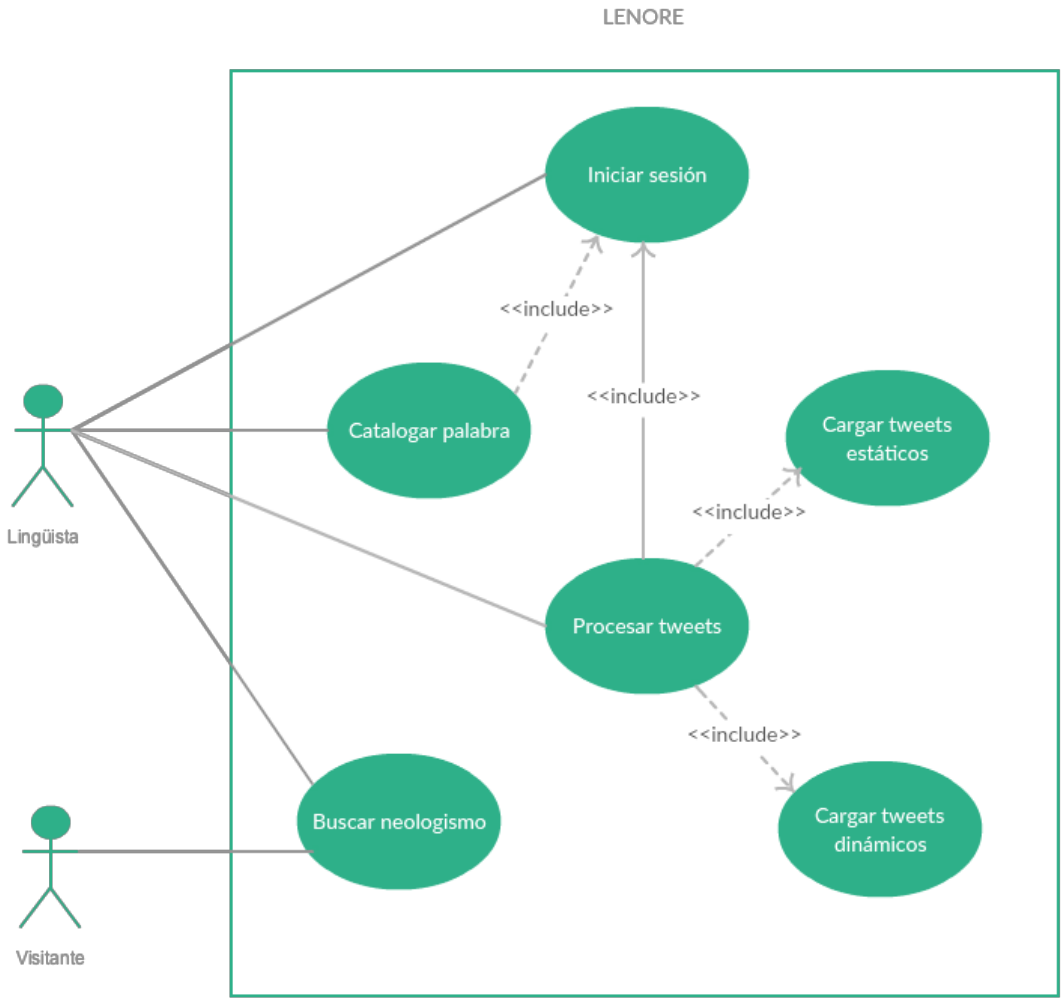


Figura 3-2: Diagrama de Casos de Uso

A continuación, se procede a detallar dos de los casos de uso más representativos del sistema.

- **CU01. Carga estática de tweets**
  Actores primarios: Usuario registrado en el sistema.
  Interesados y objetivos:
    - El lingüista quiere añadir un nuevo archivo de tweets al sistema para que sea procesado.

  Resumen: El lingüista iniciará sesión en la aplicación para poder acceder a la pestaña de Procesado. Una vez ahí seleccionará la carga estática de tweets y subirá un archivo que será procesado y añadido al sistema con sus correspondientes candidatos.

  Precondiciones: El usuario del sistema se ha autenticado como tal.

  Garantía de éxito (postcondiciones): El nuevo archivo ha sido procesado correctamente y todas las palabras candidatas y descartadas añadidas al sistema.

  Escenario principal de éxito:

  1. El usuario del sistema (ya iniciada la sesión) navega hasta la pestaña de *Procesar*.
  2. La página web muestra dos opciones, entre la que se encuentra la carga estática de tweets.
  3. El lingüista hace clic al botón de seleccionar archivo.
  4. El usuario navega hasta la ruta o localización del archivo.
  5. El usuario acepta y dicho archivo es subido al servidor a un *path* concreto.
  6. El usuario confirma la opción de procesador del archivo.
  7. El sistema empieza a analizar el archivo, tras lo que nos aparece una pantalla diciendo que se ha procesado.
  8. El lingüista hace clic en el botón que avisa de que se ha procesado y éste nos lleva a la pantalla de *Catalogar*, donde podremos visualizar las nuevas palabras que hayan sido admitidas como candidatas.

- **CU02. Admitir un candidato**
  Actores primarios: Usuario registrado en el sistema.
  Interesados y objetivos:
    - Tras el análisis del programa LECONORE sobre un tweet y encontrar varios candidatos a neologismos, el lingüista quiere que una palabra pase de candidato a admitido.

  Resumen: El lingüista iniciará sesión en la aplicación para poder acceder a la pestaña de Catalogar. Una vez, en la pestaña correspondiente de la tabla de palabras, navegará a *Candidatos* y, tras encontrar la palabra que le interesa, pulsará en el botón de Aceptar para que dicho candidato pase a la pestaña de *Admitidos*.

  Precondiciones: El usuario del sistema se ha autenticado como tal, además de existir alguna palabra candidata.

Garantía de éxito (postcondiciones): Se ha aceptado una palabra candidata actual como admitida.

Escenario principal de éxito:

1. El usuario del sistema (ya iniciada la sesión) navega hasta la pestaña de *Catalogar*.
2. La página web muestra una tabla con varias pestañas: Todos, Candidatos, Admitidos, Neologismos, Publicados y Descartados.
3. El lingüista hace clic en la pestaña de Candidatos.
4. El usuario visualiza todos los candidatos y busca manualmente o mediante la caja de texto la palabra candidata a admitir.
5. El usuario dispone de un botón para aceptar esa palabra como candidata o un botón de rechazar para convertirla en descartada.
6. El usuario hace clic en el botón de Aceptar.
7. Se muestra un mensaje por pantalla de que se ha aceptado el neologismo en concreto.

Extensiones (flujos alternativos):

El sistema no dispone de ninguna palabra candidata.

a. El sistema no mostrará ningún botón de aceptar o descartar, ya que la tabla en cuestión estará vacía. Se notifica en la propia tabla de la ausencia de datos.

4 Diseño

4.1 Introducción

Una vez realizada la revisión del estado del arte y el análisis de la aplicación, en este apartado procederemos a hacer mención de manera más profunda y justificada de las tecnologías necesarias para el desarrollo del proyecto, la arquitectura de la aplicación, su estructura, y los elementos que la componen.

4.2 Esquema de arquitectura

Tras la definición de los nuevos requisitos y mencionar los ya existentes, es el momento de hacer un resumen de la arquitectura que dará soporte a todo el proyecto. Podemos afirmar que el proyecto se encuentra dividido en dos partes diferenciadas:

- **Front-end o Interfaz de usuario**
  Es la capa encargada de la interactuación del usuario final con la aplicación y la representación y estilización de los datos que esta almacena. Incluye todas aquellas tecnologías que van a ser utilizadas desde un navegador web. Asimismo, sirve de fuente de entrada de datos para el back-end de la aplicación. Existen varias tecnologías que puedes responder a estas necesidades, entre las que encontramos HTML5, CSS, Bootstrap, JavaScript, JQuery, Ajax, etc.

- **Back-end o Entorno de procesado**
  Es la parte del proyecto destinada al procesado y al filtrado de la información que entra a la aplicación, su gestión y su almacenamiento. Las tecnologías implicadas que satisfacen dichas necesidades incluyen gestores de bases de datos SQL, NoSQL y diversas bibliotecas de programación.

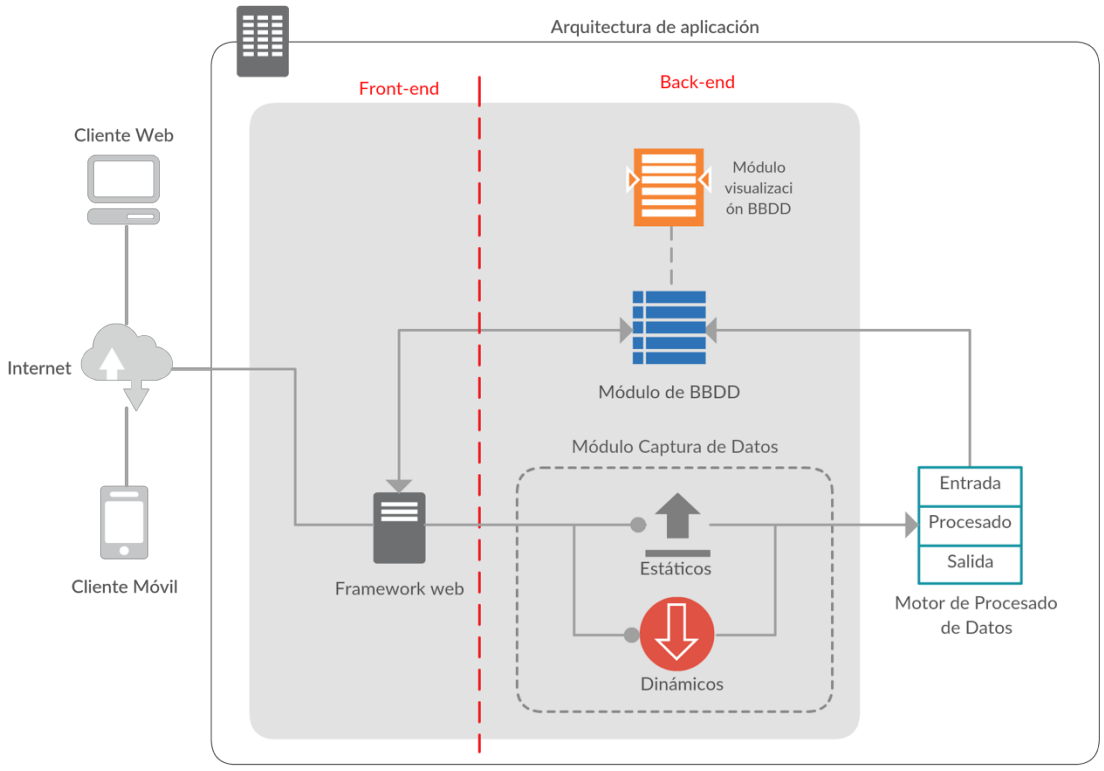


Figura 4-1: Esquema de arquitectura de la herramienta

Como se puede observar en la *figura 4-1*, los distintos elementos básicos que integrarán la aplicación en la parte de front-end serán, por un lado, un navegador web para la visualización y el framework encargado de representar la información a los usuarios en un formato legible para estos navegadores. Por otro lado, los elementos del back-end consistirán en un motor de procesamiento de archivos en bruto recibidos desde la interfaz de usuario, un módulo de base de datos que almacenará la información procesada y un módulo para visualizar esta información a nivel de desarrollador. Por último, destacamos el módulo de captura de datos que hace de enlace entre el framework web desde donde es llamado y el motor de procesado de datos. Dicho módulo de captura tendrá, a su vez, dos maneras de operar: la carga de datos estáticos o previamente descargados y la carga de datos dinámica o en tiempo real.

4.3 Elementos de la arquitectura

Tras hacer referencia a la arquitectura que seguirá la aplicación en la *figura 4-1*, vamos a mencionar cada uno de los elementos que componen las dos partes:

- **Framework web**

  Este módulo es el más importante ya que hace de enlace entre los distintos elementos de la aplicación. Se encarga de representar la página web con todos sus componentes, del funcionamiento y lógica de la aplicación y obtener la información necesaria de la base de datos en cuestión. Entre los frameworks de desarrollo más populares podemos encontrar Angular, Meteor, Ruby on Rails, Laravel o Django.

- **Módulo de Base de Datos**

  Este componente es crítico ya que en él se encuentra almacenada la información que es utilizada en la aplicación. Con el paso del tiempo, esta base de datos recogerá una gran cantidad de información, por lo que es imperativo encontrar una tecnología que permita guardar y consultar texto de manera eficiente. Disponemos de opciones variadas en función de nuestras necesidades: existen *bases de datos relacionales*, como PostgreSQL, MySQL o SQL Server, que utilizan el modelo relacional y dependen fuertemente en la consistencia de los datos, mientras que las *bases de datos no relacionales*, como MongoDB, Apache Solr o Elasticsearch, tratan de solventar algunas de las limitaciones de este modelo cuando los datos a almacenar no son estructurados.

- **Módulo de visualización de datos**

  Existen diversas herramientas que nos ayudan a presentar la información que almacenamos previamente en nuestras bases de datos de una manera clara, cercana y que llame la atención. Además, sirven de ayuda y apoyo durante el desarrollo ya que permiten realizar comprobaciones del correcto funcionamiento de la aplicación y coherencia de los datos.

- **Módulo de Captura de Datos**

  Este módulo sirve de conexión entre el framework web y el motor de procesado de datos. El objetivo de este elemento es ofrecer al usuario la posibilidad de escoger entre diferentes opciones para suministrar una fuente de datos a analizar para la herramienta.

- **Motor de Procesado de Datos**

  Es el motor de procesamiento de archivos en formato JSON con los tuits en bruto a analizar. El Procesamiento del Lenguaje Natural es una rama de conocimiento que pretende conseguir que una máquina entienda lo que una persona quiere decir con lenguaje natural. Su funcionalidad es la de descubrir palabras bien formadas en castellano que hayan pasado los diferentes filtros de criba. Para ello, cuando se analizan los tuits se normalizan las letras, se trata de eliminar hashtags, emoticonos, menciones, saltos de línea, etc. Tras pasar por todos los filtros, las palabras resultantes serán las candidatas a ser neologismo, pero esto último lo decidirá el lingüista que haga uso de la aplicación tras ser añadidas de nuevo al sistema.

Todo lo anteriormente expuesto podemos comprenderlo de una manera más clara en la *figura 4-1*, donde podemos visualizar cada uno de los elementos implicados en la arquitectura.

5 Desarrollo

5.1 Introducción

El propósito de este capítulo es el de listar y mencionar de manera más detallada los elementos de la arquitectura, así como los elementos escogidos en las diferentes partes que componen el proyecto. Asimismo, se explicarán las funcionalidades acompañadas de las correspondientes capturas de pantalla que ilustren los flujos de navegación y diseño de la aplicación.

5.2 Arquitectura de LECONORE

Tras el análisis de la arquitectura de la aplicación y las diferentes tecnologías implicadas, se detalla a continuación las decisiones tomadas para la implementación. En la *figura 5-1* podremos observar de manera más concreta los elementos que han sido seleccionados. De las dos partes anteriormente mencionadas en la arquitectura:

- **Front-end o Interfaz de usuario**
  Para la implementación de esta capa encargada de la interactuación del usuario final con la aplicación se ha decidido utilizar el framework de Django, el lenguaje de programación Python, y HTML5, CSS y Bootstrap para el diseño web de la aplicación.

- **Back-end o Entorno de procesado**
  Las tecnologías seleccionadas para el entorno de procesado basándose en criterios técnicos son Elasticsearch, Kibana y diferentes módulos proporcionados por el Laboratorio de Lingüística Informática de la UAM, encargados de transformar los requisitos funcionales puramente lingüísticos a un código de programación equivalente.

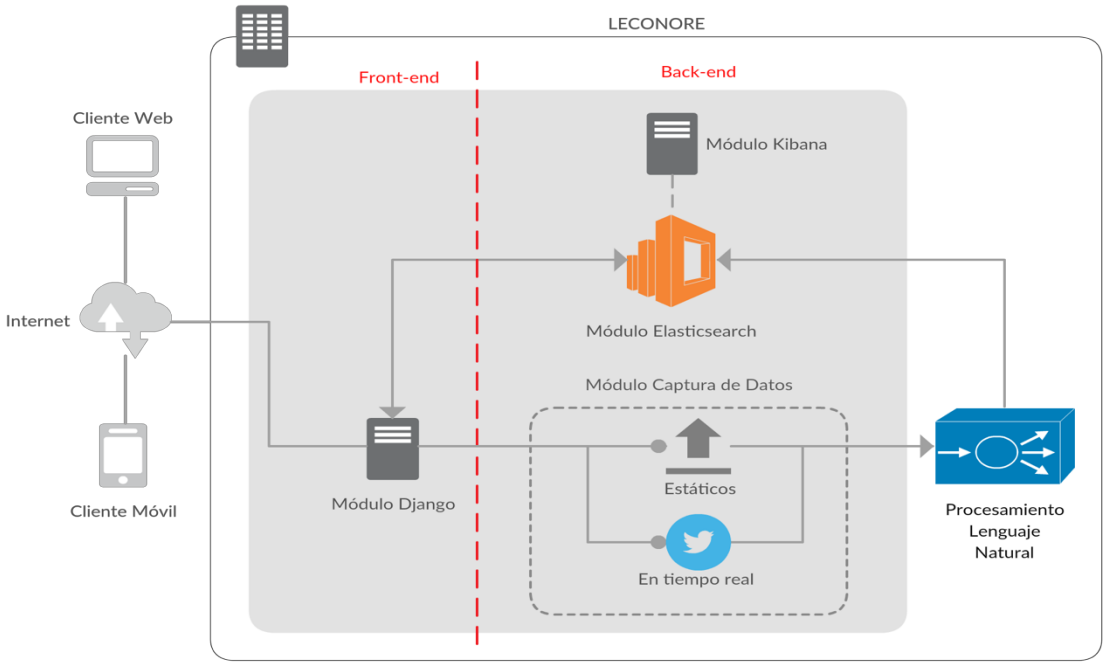


Figura 5-1: Implementación de la arquitectura específica de LECONORE

A su vez, en la *figura 5-2* se desglosa el módulo de Procesamiento de Lenguaje Natural de la *figura 5-1* en tres partes: extracción, PLN y GRAMPAL.

Figura 5-2: Desglose Procesamiento Lenguaje Natural

5.3 Elementos de LECONORE

Después de referenciar la arquitectura que seguirá LECONORE en la *figura 5-1*, se detallan los elementos que componen el front-end y back-end a continuación:

- **Módulo Django**

  Este servidor es el más importante ya que hace de enlace entre los distintos módulos de la aplicación, gracias al modelo MTV y las distintas APIs utilizadas. Se encarga de representar la página web con todos sus componentes, del funcionamiento y lógica de la aplicación y obtener la información necesaria de la base de datos pertinente. En el *apartado 5.7* se procede a dar más detalle acerca de cómo es representada la información.

- **Módulo Elasticsearch**

  En LECONORE, este componente es crítico ya que en él se encuentran los distintos índices que almacenan la información de las palabras candidatas, admitidas, descartadas, neologismos y publicados. Actualmente disponemos de la versión 6.4.1, que podremos visualizar si accedemos a la URL y puerto donde está ejecutándose la instancia.

```
{
  "name" : "9DKX-zf",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "EDLDePruTD6oiilXh8Wizw",
  "version" : {
    "number" : "6.4.1",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "e36acdb",
    "build_date" : "2018-09-13T22:18:07.696808Z",
    "build_snapshot" : false,
    "lucene_version" : "7.4.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

  La manera de conectar el servidor de Elasticsearch con nuestro framework de desarrollo ha sido haciendo uso de una biblioteca o cliente a bajo nivel llamado *Elasticsearch-py* [1]. La finalidad de esta API es la creación de una herramienta común que gestione las peticiones de Elasticsearch escrita en Python.

- **Módulo Kibana**

  Utilizado de manera informativa y como apoyo durante el desarrollo del proyecto, provee información útil a nivel de desarrollo. No es imprescindible que la instancia del servidor se encuentre operativa y disponible de manera permanente.

- **Módulo de Captura de Datos**

  Previamente a la realización de este TFG, el módulo de captura de datos ya disponía de una funcionalidad que permite, a través de la interfaz de usuario, la carga de archivos JSON con información de tuits descargados con anterioridad para subirlos a la herramienta. Adicionalmente, para este trabajo se ha definido un requisito funcional que permite la descarga de tuits en tiempo real haciendo uso de la API de Twitter. En *apartado 5.5* que podemos encontrar más abajo se procede a detallar de manera más técnica las soluciones y decisiones tomadas en la implementación de esta funcionalidad añadida.

- **Motor PLN o Procesamiento de Lenguaje Natural**

  Este motor de procesamiento de archivos recibe la información desde módulo de Captura de Datos para realizar un tratamiento de esta en tres fases. Posteriormente los datos son enviados al módulo de Elasticsearch para introducirlos en la base de

datos. El procedimiento de Procesamiento de Lenguaje Natural se detalla más en profundidad en el *apartado 5.6*

5.4 Estructura en árbol

La estructura en árbol que toman los menús de la página web tras su rediseño y los elementos queda detalladas a continuación:

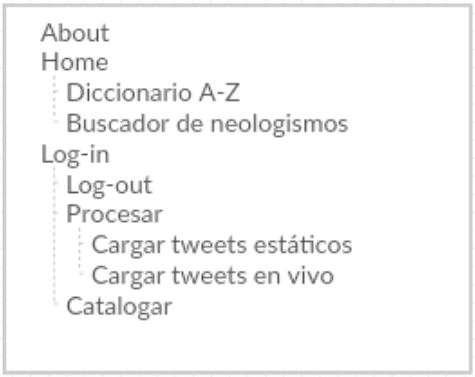


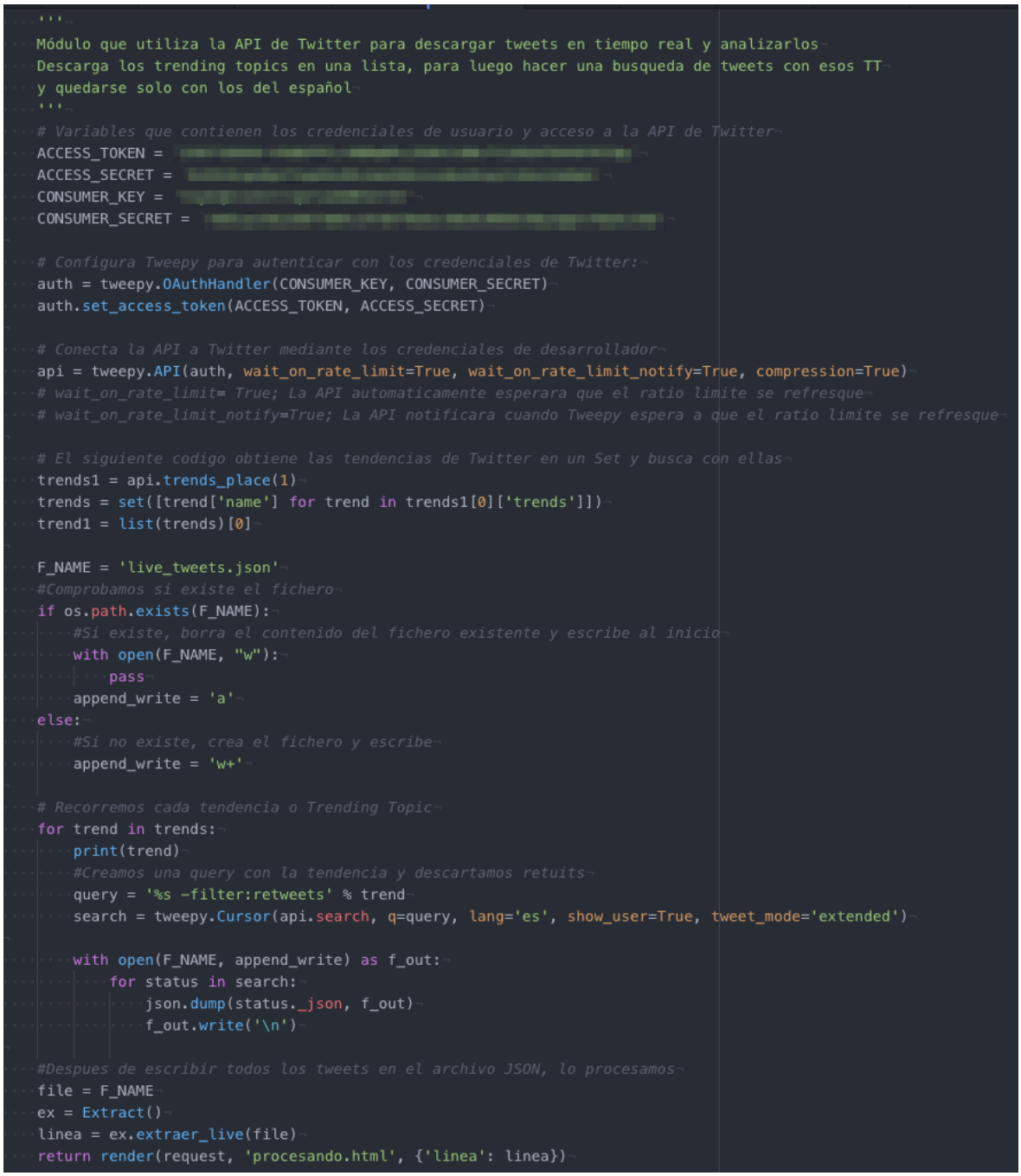
Figura 5-3: Mapa de árbol web LECONORE

Como se puede observar en la *figura 5-3*, la página carga por defecto en *About*, donde visualizaremos una introducción a la aplicación, en qué consiste, las entidades y personas involucradas, así como un pequeño apartado con estadísticas y formulario para contactar con el departamento. Dispondremos también de la pestaña *Home*, donde podremos ver detalles acerca de neologismos añadidos recientemente o realizar búsquedas de palabras ya añadidas al sistema. Por último, la pestaña de *Log-in* permitirá a los usuarios administradores iniciar sesión para desbloquear funcionalidades extra como *Catalogar* o *Procesar* tweets, subdividido a su vez en una estática de tweets y una lectura en vivo.

5.5 Descarga de tuits en tiempo real

En este apartado se detallan los pasos seguidos para la implementación del módulo que permite la descarga de tuits en tiempo real y diversas consideraciones a tener en cuenta durante el desarrollo de dicha funcionalidad.

En primer lugar, se ha de solicitar la activación de la cuenta de Twitter para desarrolladores rellenando unos formularios donde se detalla el uso específico que se va a hacer de los datos. Teniendo en cuenta las limitaciones de la API de Twitter con relación al número de tuits que se pueden obtener, a nivel de interfaz gráfica se ha decidido implementar un botón (*figura 5-8*) para pulsar cada vez que queramos obtener más candidatos en lugar de disponer de un interruptor que mantuviera apagada o encendida permanentemente esta función.



```
'''
Módulo que utiliza la API de Twitter para descargar tweets en tiempo real y analizarlos
Descarga los trending topics en una lista, para luego hacer una busqueda de tweets con esos TT
y quedarse solo con los del español
'''
# Variables que contienen los credenciales de usuario y acceso a la API de Twitter
ACCESS_TOKEN =
ACCESS_SECRET =
CONSUMER_KEY =
CONSUMER_SECRET =

# Configura Tweepy para autenticar con los credenciales de Twitter:
auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)

# Conecta la API a Twitter mediante los credenciales de desarrollador
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True, compression=True)
# wait_on_rate_limit= True; La API automaticamente esperara que el ratio limite se refresque
# wait_on_rate_limit_notify=True; La API notificara cuando Tweepy espera a que el ratio limite se refresque

# El siguiente codigo obtiene las tendencias de Twitter en un Set y busca con ellas
trends1 = api.trends_place(1)
trends = set([trend['name'] for trend in trends1[0]['trends']])
trend1 = list(trends)[0]

F_NAME = 'live_tweets.json'
#Comprobamos si existe el fichero
if os.path.exists(F_NAME):
    #Si existe, borra el contenido del fichero existente y escribe al inicio
    with open(F_NAME, "w"):
        pass
    append_write = 'a'
else:
    #Si no existe, crea el fichero y escribe
    append_write = 'w+'

# Recorremos cada tendencia o Trending Topic
for trend in trends:
    print(trend)
    #Creamos una query con la tendencia y descartamos retuits
    query = '%s -filter:retweets' % trend
    search = tweepy.Cursor(api.search, q=query, lang='es', show_user=True, tweet_mode='extended')

    with open(F_NAME, append_write) as f_out:
        for status in search:
            json.dump(status._json, f_out)
            f_out.write('\n')

#Despues de escribir todos los tweets en el archivo JSON, lo procesamos
file = F_NAME
ex = Extract()
linea = ex.extraer_live(file)
return render(request, 'procesando.html', {'linea': linea})
```

Figura 5-4: Módulo Live Tweets

Como podemos observar en la *figura 5-4*, el procedimiento para descargar tuits en tiempo real es el siguiente:

- Una vez obtenidos los tokens y claves de desarrollador, podemos especificar en Tweepy ciertos argumentos en la función de conexión a la API para que LECONORE automáticamente espere a que la ratio límite de tuits se reinicie y/o se notifique por ello.

- Después de autenticar se obtienen los Trending Topics a nivel mundial y se comprueba la existencia del fichero donde se van a volcar los datos. En caso de

existir, primero se vaciará para evitar un cúmulo excesivo de datos, en caso contrario se creará el archivo.

- Se procede a crear una query con cada una de las tendencias globales de Twitter, se descartan retuits, se especifica mediante otro argumento de la función de búsqueda el idioma y se la devolución del cuerpo del tuit completo.

- Tras la búsqueda y volcado de información JSON, se trata de extraer la información relevante del fichero.

Como la API devuelve los tuits en un formato diferente al de los tuits descargados anteriormente, ha sido necesario además modificar el módulo de extracción para que atendiera a distintos criterios en función de la procedencia de estos ficheros.

5.6 Extracción, filtros PLN y GRAMPAL

El Procesamiento de Lenguaje Natural es, como se mencionó en la arquitectura, un módulo crítico para la herramienta LECONORE. De él depende la funcionalidad de extracción de neologismos a partir de tuits descargados y está dividido en tres partes:

- La parte de **extracción** está escrita en Python y se encarga de leer el archivo JSON recibido por el módulo de Captura de Datos para obtener los campos que interesan de cada tuit. Al conocer la estructura que define cada tuit, se procede a leer cada línea del fichero y seleccionar los campos relevantes como son su ID, la biografía del usuario, la localización, la fecha y el cuerpo del mensaje.

- La sección de PLN consiste en una variedad de filtros que desempeñan diversas tareas de limpieza del texto resultante. Se procede a limpiar caracteres específicos como arrobas, enlaces, números romanos, saltos de línea, hashtags, letras repetidas en exceso, diminutivos y clíticos. La salida de cada uno de estos filtros sirve para alimentar el siguiente. Finalmente se pasa un último filtro consistente en detectar un patrón de palabras sueltas para obtener un lexema que pasará a la siguiente fase de filtrado.

- En el último paso, GRAMPAL recibe los lexemas 'limpios' para analizarlos. La utilidad de esta herramienta es realizar un análisis morfosintáctico de las palabras recibidas para detectar la función que estas desempeñan. En el caso de que no sean reconocidas por GRAMPAL, significa que estas palabras son desconocidas y, por lo tanto, fuertes candidatos a ser un neologismo.

Tras la revisión de requisitos funcionales definidos en el *apartado 3.4.1* por parte de los integrantes del departamento de Lingüística Informática, se llegó a la conclusión de que convendría realizar algunas modificaciones para obtener un mayor número de candidatos a neologismo, detalladas a continuación:

- Para la **extracción** de campos de tuits descargados en tiempo real ha sido necesario rediseñar la función de extracción para adaptarla a los nuevos campos que proporciona la API de Tweepy, ya que difieren de los tuits descargados previamente.

- En cuanto a **PLN**, se ha decidido prescindir del filtro concreto que detectaba clíticos, es decir, no se realizan combinaciones de dos listas de sufijos (["lo", "la", "le", "les"] y ["me", "te", "se"]) para detectarlos en palabras filtradas y eliminarlos.

- En el analizador morfosintáctico **GRAMPAL**, encargado de conocer qué categoría sintáctica corresponde a una palabra, se evita despreciar candidatos que parezcan diminutivos y clíticos.

5.7 Interfaz web

En este apartado se hará mención de las características y funcionalidades principales de la aplicación acompañadas de una imagen que muestra cómo luce la página web.

5.7.1 About Us

Como podemos observar en la *figura 5-5*, se hace una breve introducción al proyecto y sus funcionalidades, además de mencionar a los integrantes involucrados en el desarrollo de la herramienta. También se ha incluido un formulario de contacto al final de la página.

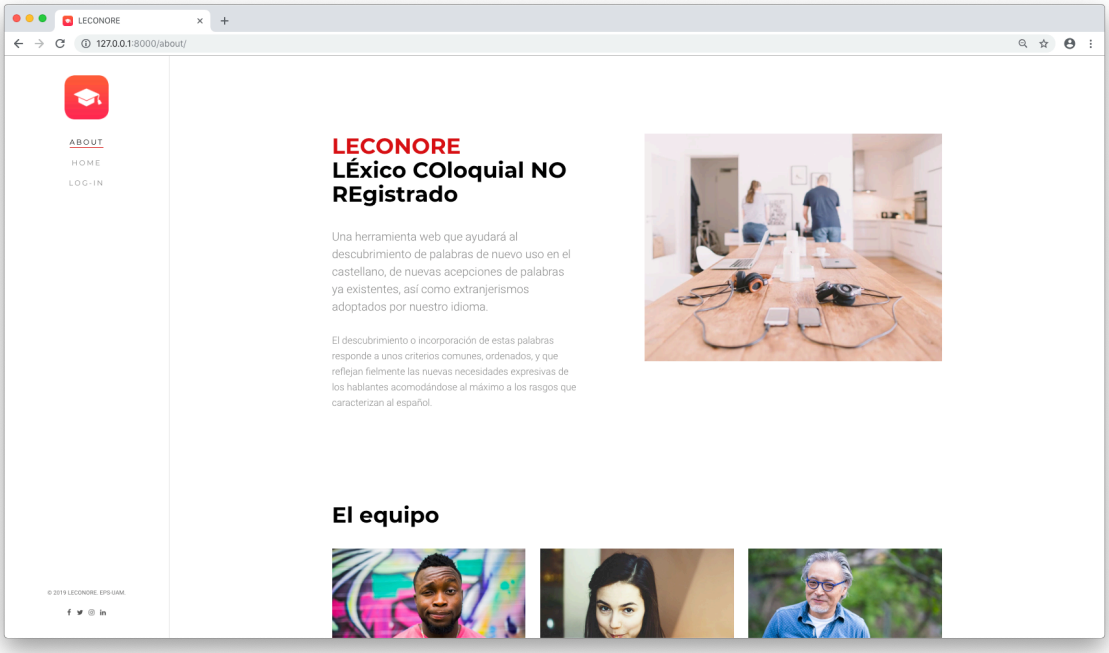


Figura 5-5: Pantalla About Us

5.7.2 Home

En la pestaña Home, mostrada en la *figura 5-6*, se puede ver el diccionario de neologismos. Esto nos permite, de un vistazo, saber si existe algún neologismo que comience específicamente por alguna de las letras del abecedario, en este caso por la letra A, B, E, S

y T. Además, es posible visualizar los últimos neologismos añadidos con paginación, de manera que acortamos la longitud de la página añadiendo funcionalidad.

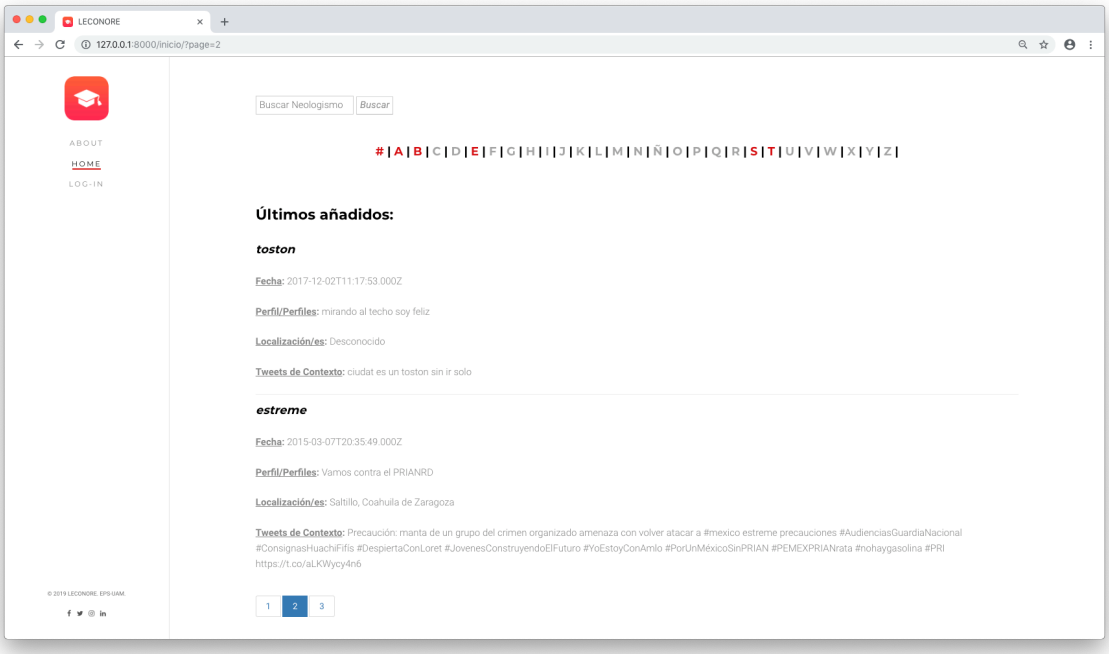


Figura 5-6: Home – Últimos Añadidos, Diccionario y Paginación

La funcionalidad del diccionario de neologismos por letra puede verse más en detalle en la *figura 4-7*. Tras seleccionar la letra *S*, veremos en color rojo *spot*, el único neologismo que comienza por dicha letra junto con información contextual como el perfil del usuario que la utilizó, la fecha, la localización y un tuit en el que fue utilizada.

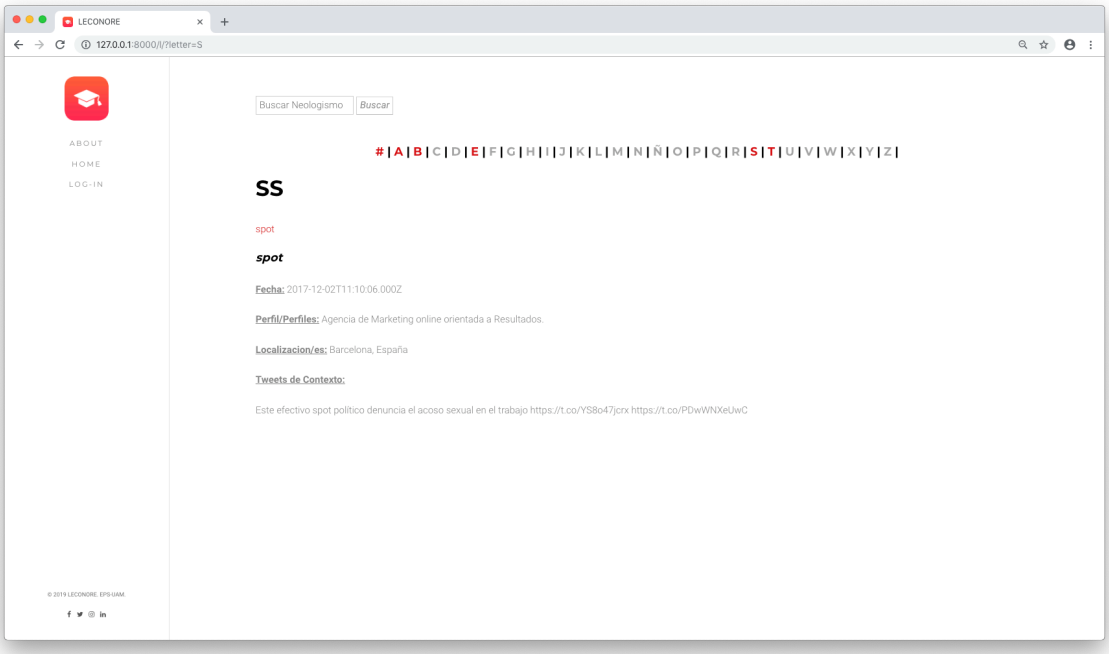


Figura 5-7: Diccionario por Letra

Como vemos en la *figura 5-8*, al final de la página se dispone de un recuento de datos de manera informativa, independientemente del rol del usuario.

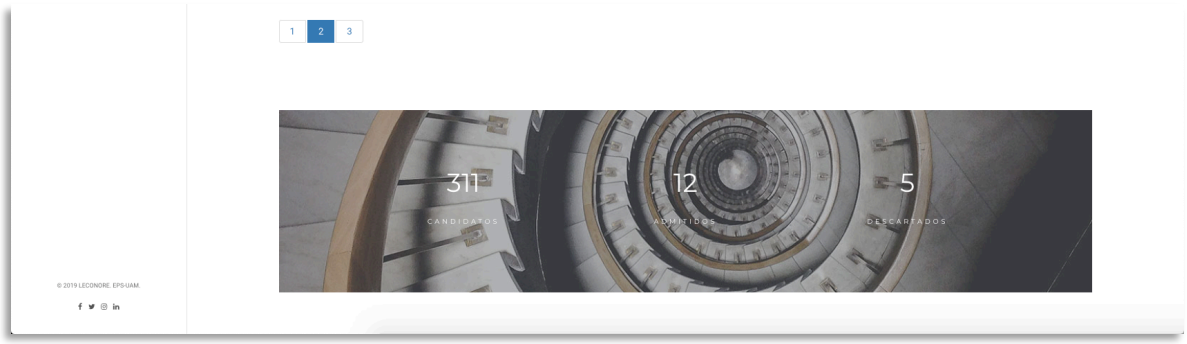


Figura 5-8: Home – Estadísticas

5.7.3 Log-in

A continuación, en la *figura 5-9* se muestra la pantalla correspondiente que permite iniciar sesión en la aplicación LECONORE y pasar a obtener funcionalidades de administrador. Una vez se inicia sesión correctamente, el usuario es redirigido a la página de *About Us*, desde donde se observa un botón para cerrar sesión y la aparición de dos nuevas pestañas en el menú: *Procesar* y *Catalogar*.

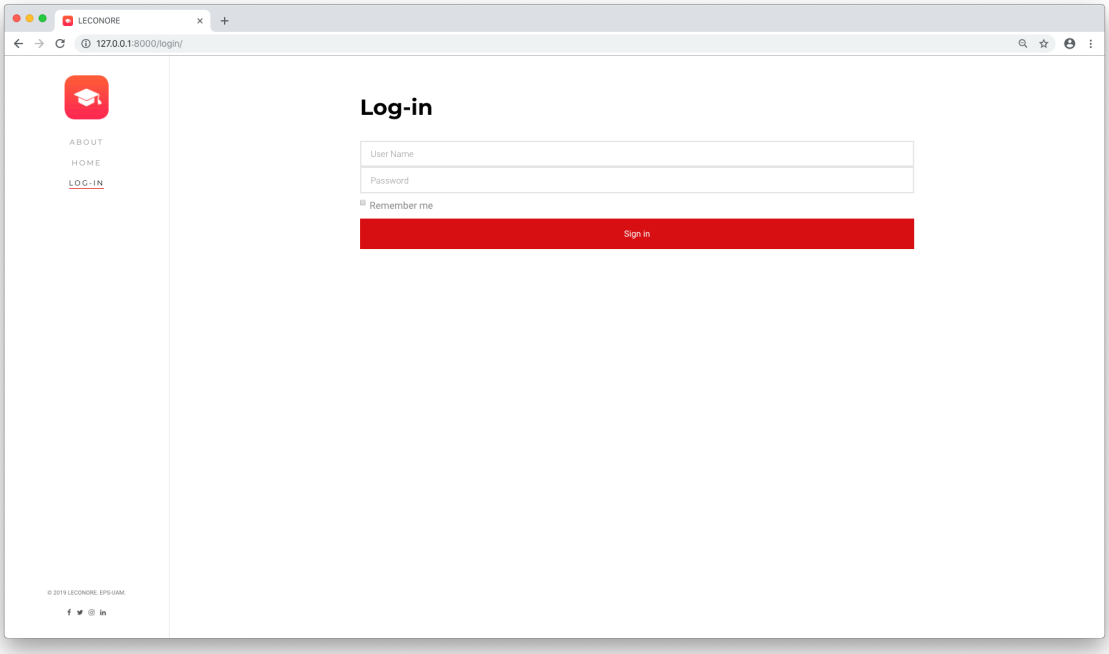


Figura 5-9: Log-in

5.7.4 Procesar

Esta pestaña, mostrada en la *figura 5-10*, tiene una nueva funcionalidad con respecto al anterior TFG sobre el que está basado el presente documento: además de disponer de la carga estática de ficheros, se ha creado un botón que descarga tuits en tiempo real haciendo uso de la API de Twitter.

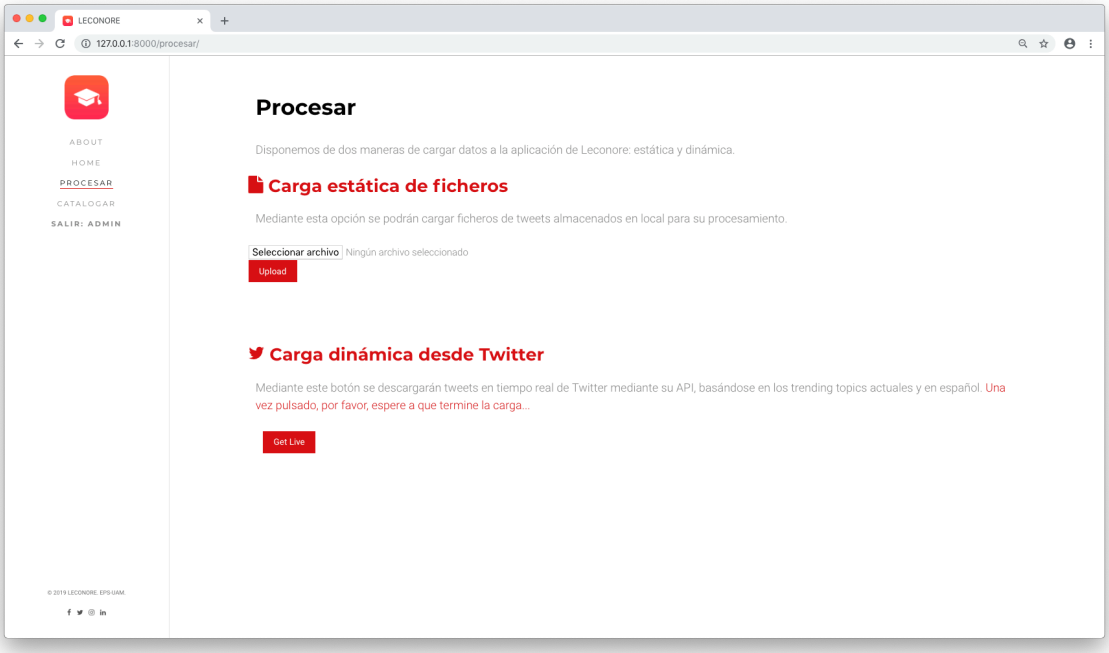


Figura 5-10: Procesar

5.7.5 Catalogar

Este apartado es de vital importancia para los administradores del sistema, ya que permitirá a dichos usuarios catalogar las palabras que previamente han sido cargadas al sistema desde la pestaña de *Procesar*.

Las pestañas de color rojo permiten clasificar las distintas palabras en función de diversos criterios mencionados a continuación: si son candidatos a neologismo, si han sido admitidas como candidato, si son descartadas como candidato a neologismo, si han sido catalogadas definitivamente como neologismo o si dichos neologismos han sido publicados en LECONORE.

Como podemos observar en la *figura 5-11*, es posible visualizar todos los términos que han entrado al sistema, independientemente de su clasificación: incluye los candidatos, admitidos, neologismos, publicados y descartados en la misma tabla.

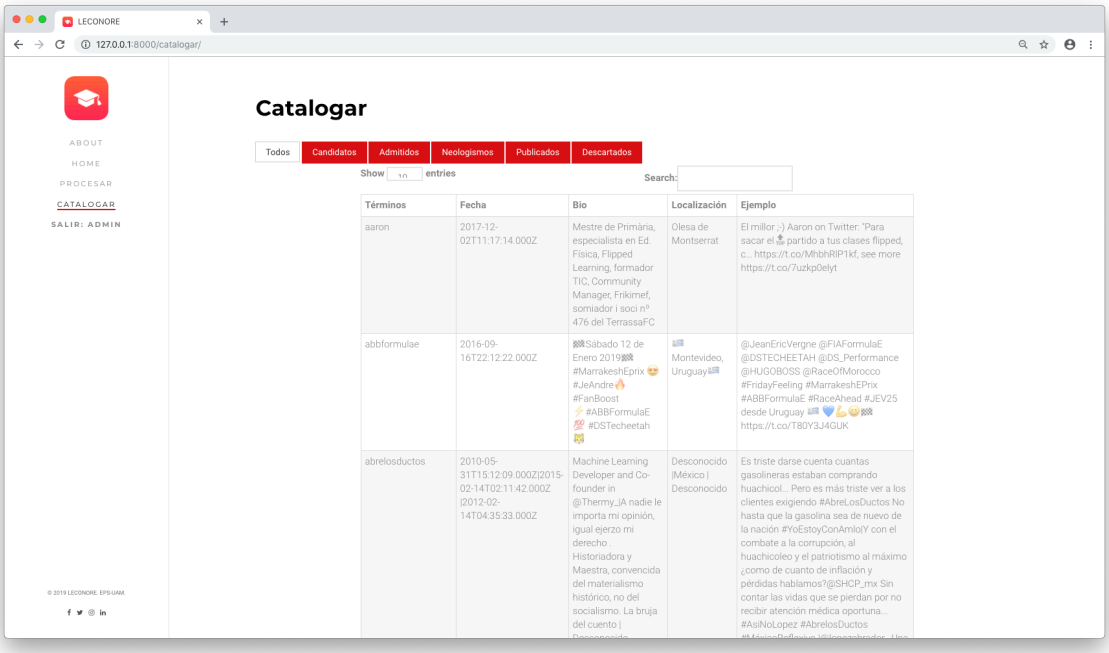


Figura 5-11: Catalogar - Todos

En las siguientes capturas de pantalla es posible observar que, una vez nos encontramos en distintas pestañas de la tabla, podemos realizar diferentes acciones. En la *figura 5-12* se visualizan todos los candidatos que, como se indica en la *figura 3-1*, pueden ser admitidos o descartados.

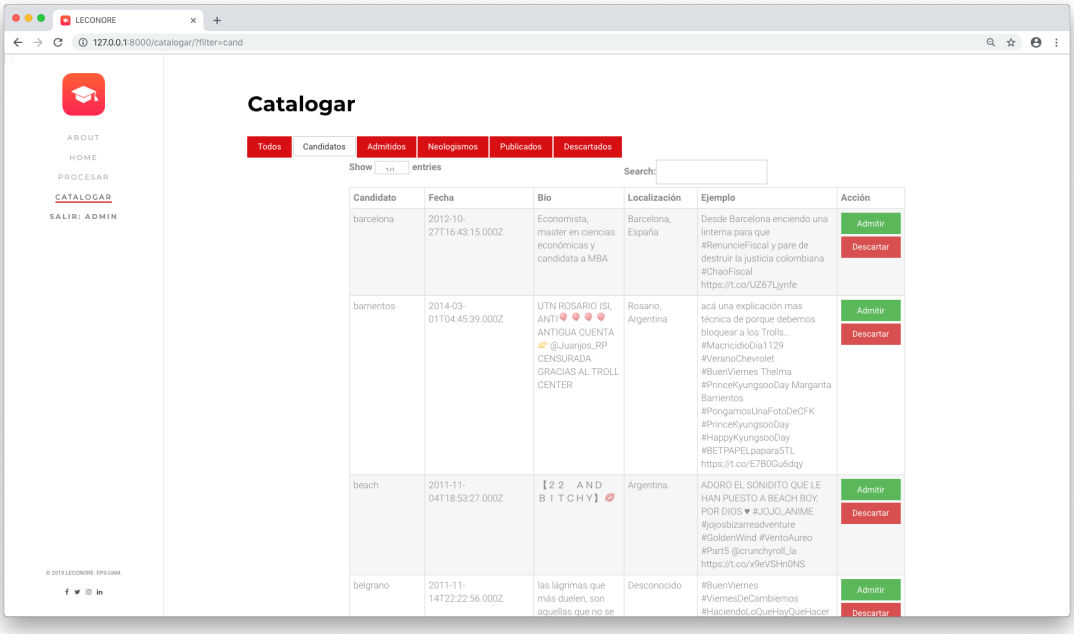


Figura 5-12: Catalogar - Candidatos

Basándose en el esquema anteriormente mencionado, en la *figura 5-13* podemos observar que un término ya admitido tiene, según la imagen, tres opciones: *lematizar* el término admitido, degradarlo a candidato o descartarlo, lo que haría que dicho término fuese moviéndose entre las pestañas de la tabla según la opción escogida.

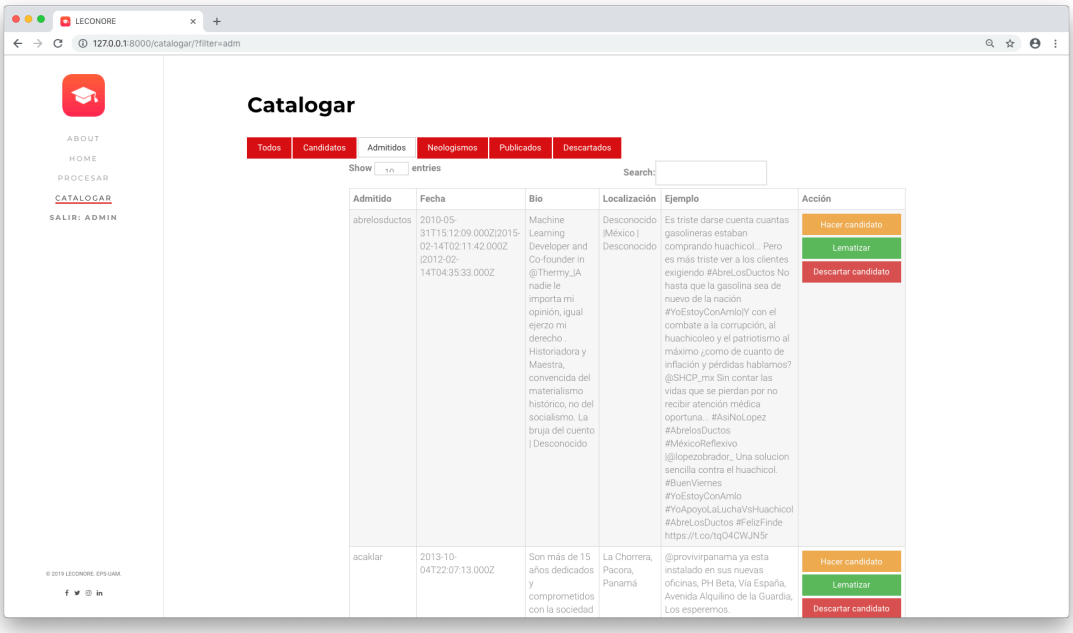


Figura 5-13: Catalogar - Admitidos

5.7.6 Interfaz responsiva y dispositivos móviles

Como se puede observar en la *figura 5-14*, la interfaz de usuario desde navegadores convencionales sufre una adaptación según las dimensiones de la página. Este rediseño responde al concepto de diseño web adaptable, o *responsive design*.

Asimismo, en las *figuras 5-15 y 5-16* se muestra la organización y diseño que toma la aplicación desde un dispositivo móvil y el funcionamiento del menú que se oculta según las necesidades del usuario.

Figura 5-14: Interfaz responsive

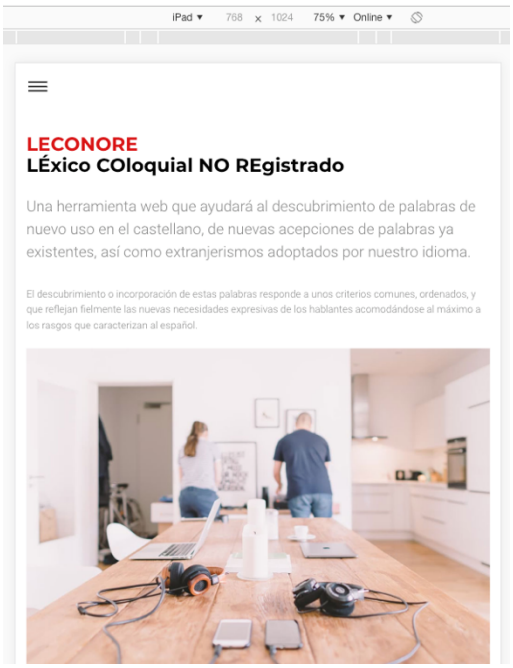


Figura 5-15: Visualización móvil

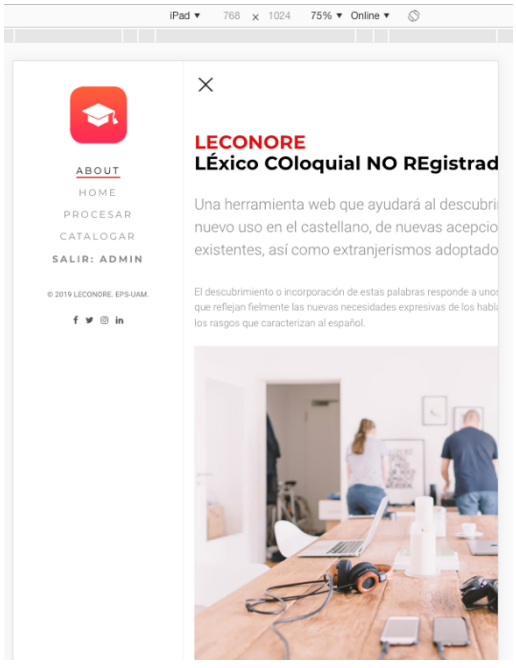


Figura 5-16: Menú móvil desplegado

6 Integración, pruebas y resultados

6.1 Integración

En primer lugar, de cara a la integración de todos los módulos involucrados en el proyecto cabe mencionar que el peso de esta integración recayó, sobretodo, en el TFG anterior.

Inicialmente fue necesario unir todos los componentes informáticos que por separado no cumplían ninguna función. Utilizando el framework de Django como esqueleto, se añadieron diversas bibliotecas que actúan de punto de unión del servidor de aplicación con Elasticsearch y el resto de los elementos necesarios en el desarrollo del proyecto.

Por nuestra parte ha sido necesario integrar la API de Twitter en la herramienta LECONORE, creando una cuenta en la red social y pidiendo el perfil de desarrollador que nos concedieron sin demora. Una vez hecho esto, y siguiendo la propia documentación del desarrollador, no fue difícil crear en código lo que nuestros requisitos funcionales demandaban.

6.2 Pruebas

A la hora de realizar las pruebas pertinentes de las nuevas funcionalidades, destacamos el uso de un sistema de registro de eventos para LECONORE desarrollado para el anterior TFG, donde se deja constancia del procesado de archivos JSON en el sistema.

Tras la descarga inicial del proyecto original para su continuación, sí que encontramos algún problema al no poder replicar el comportamiento esperado entre la instancia en local y la del servidor de producción. Finalmente se encontró en que existía cierto parámetro de configuración en Elasticsearch que había sido introducido por medio de Kibana, en vez de estar reflejado a nivel de código: se devolvía un número bastante limitado de resultados cuando hacíamos una consulta, por lo que parecía que la base de datos era incompleta.

Como podemos observar en las *figuras 6-1 y 6-2*, el dato que visualizamos en Kibana con relación al número de hits cuando aplicamos el filtro "_index is admitidos" se corresponde con la información que muestra en LECONORE.

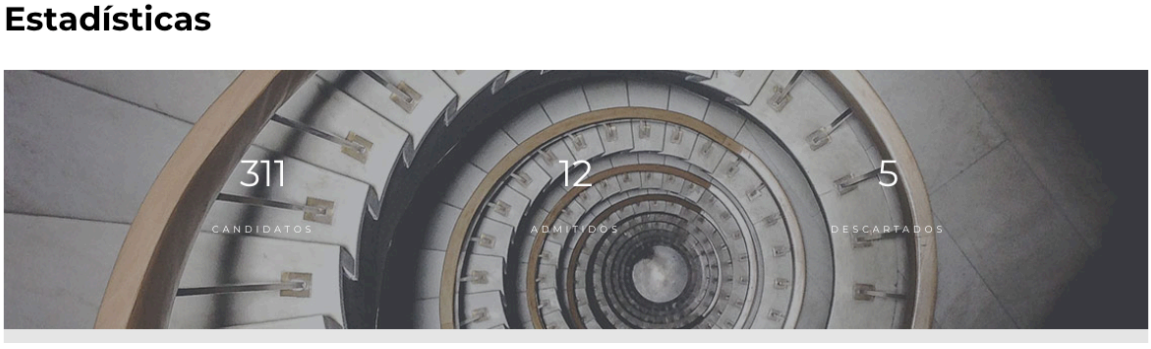


Figura 6-1: Información Front-end LENORE

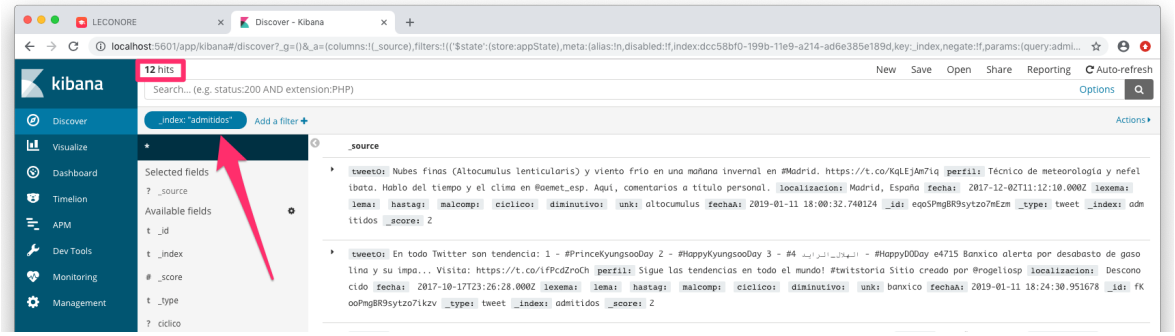


Figura 6-2: Información Back-end Kibana

Asimismo, tras realizar cambios a nivel de código fuente de la aplicación, ha sido probada para asegurar el funcionamiento esperado. Todos los fallos descubiertos y detalles a mejorar fueron anotados para realizar un seguimiento más allá de la funcionalidad requerida.

7 Conclusiones, discusión y trabajo futuro

7.1 Conclusiones

Tras realizar una valoración final una vez que el proyecto ha sido terminado, teniendo en cuenta el estado de partida de este, consideraría que el desarrollo de este TFG ha cubierto la mayoría de los objetivos fijados por las partes involucradas en el mismo.

Por un lado, basar un Trabajo de Fin de Grado en la continuación, mejora y mantenimiento y desarrollo de otro previo ha sido más desafiante de lo esperado. A priori es una tarea asequible, pero una vez tratas de profundizar para modificar o añadir funcionalidades te das cuenta de lo enrevesado o poco claro que puede resultarle a alguien su propio código, no digamos ya el ajeno.

Por otra parte, me considero una persona a la que le gusta cuidar los detalles. Esto no es malo en la mayoría de las ocasiones, pero a veces me he obcecado demasiado en intentar pulir algún detalle innecesario de la interfaz gráfica que ya cumplía holgadamente los objetivos. Aún así, la parte con la que más he disfrutado ha sin duda en el rediseño gráfico.

Para finalizar, me gustaría mencionar que este TFG ha contado con la totalidad de mi interés ya que los idiomas y la lingüística siempre han sido una rama del conocimiento por la que siento cierta propensión. Todo ello, junto al uso de una API de Twitter, me ha ayudado a sentirme más satisfecho por haber contribuido al desarrollo de una herramienta en línea con una finalidad y utilidad clara. Asimismo, me hubiera gustado conocer al resto de personas involucradas en el desarrollo de LECONORE y poder aprender más de ellos, más allá de los mails intercambiados.

7.2 Trabajo futuro

Personalmente considero que el desarrollo de este TFG aún tiene un buen recorrido por delante, incluso pudiendo servir como Trabajo de Fin de Grado para otras personas que estén dispuestas a continuar con su desarrollo.

Tras haber estado trabajando con la herramienta durante estos últimos meses, han surgido ciertas ideas que, bien por falta de tiempo, complejidad, o alcance, no han sido implantadas. Dichas sugerencias podrían mejorar exponencialmente el rendimiento y mantenimiento de la herramienta, entre ellas:

- **Documentar y ordenar el código** acorde a las guías de estilo de Python, de cara a que cualquier persona con algo de base de programación pueda encontrar de manera sencilla y directa la línea que hay que tocar, entender el funcionamiento general de una función concreta, diferenciar entre una de las tantas variables del sistema o la razón de existir de una clase, entre otras.

- Tras recabar más información acerca de Elasticsearch y analizar el proyecto en sus orígenes, llegamos a la conclusión de que se están utilizando muchos índices para gestionar los datos de los posibles neologismos. En concreto, la mejora propuesta

consiste en el **uso de un único índice Elasticsearch** que tenga un atributo que diferencie entre candidatos, admitidos, descartados, neologismos y publicados, ya que cada vez que movemos vocabulario de un índice a otro estamos destruyendo y creando datos, con la consiguiente carga computacional. Para realizar este cambio hay que modificar una cantidad notable de funciones, pero una vez realizado el código pasará a ser más legible y el rendimiento mejorará notablemente.

- **Afinar los filtros del extractor** ya que actualmente son usados pocos campos por tuit, disponiendo de bastante información con la que poder montar un *dashboard* o panel de estadísticas más complejo.

- **Continuar con la mejora de la interfaz gráfica**, adaptándola a nuevas funcionalidades que vayan surgiendo.

- **Creación de filtros PLN más complejos**, en colaboración con el departamento de Lingüística, que eviten palabras extranjeras de uso común que pasan por el módulo PLN aun estando el tuit en castellano.

Referencias

[1] El español es la segunda lengua en Twitter – El País – Madrid, 14/01/2013 https://elpais.com/cultura/2013/01/14/actualidad/1358165374_278922.html
[2] https://developer.twitter.com/en/docs.
[3] http://www.tweepy.org/
[4] https://www.elastic.co/guide/en/elasticsearch/reference/6.2/_basic_concepts.html
[5] https://docs.python.org/3/
[6] https://djangobook.com/mdj2-django-structure/
[7] https://elasticsearch-py.readthedocs.io/en/master/api.html

Glosario

API	Application Programming Interface
API Restful	API que utiliza peticiones HTTP como GET, PUT, POST y DELETE
RAE	Real Academia Española
PLN	Procesamiento de Lenguaje Natural
LECONORE	Lexico Coloquial No Registrado
TFG	Trabajo de Fin de Grado
TL	Time Line
TT	Trending Topic
HTTP	HyperText Transfer Protocol
CSS	Cascade StyleSheet
SQL	Structured Query Language
NoSQL	Not only Structured Query Language
GNU	GNU is Not Unix
IDLE	Integrated DeveLopment Environment
MTV	Model Template View
MVC	Model View Controller
BBDD	Bases de Datos
W3C	World Wide Web Consortium
RF	Requisito Funcional
RNF	Requisito No Funcional
CU	Caso de Uso
UAM	Universidad Autónoma de Madrid
URL	Uniform Resource Locator
Clítico	Elemento grammatical escrito como una palabra o partícula átona independiente, pero que en realidad se pronuncia como parte de la palabra anterior o siguiente

Anexos

A Tweet en formato JSON de Tweepy


```
{
 "created_at": "Sat Dec 29 11:04:20 +0000 2018",
 "id": 1078969965800316928,
 "id_str": "1078969965800316928",
 "full_text": "\u201cLa cocina siempre ha estado en el plano de lo sensorial, pero filtrada por una cultura. En funci\u00f3n de eso, cada uno ha aprendido lo que est\u00e1 bueno y eso es algo indiscutible para el noventa por ciento de la poblaci\u00f3n.\u201d, Andoni Luis Aduriz https://t.co/woDxrz42fp",
 "truncated": false,
 "display_text_range": [0, 239],
 "entities": {
  "hashtags": [],
  "symbols": [],
  "user_mentions": [],
  "urls": [],
  "media": [
   {
    "id": 1078969957327798272,
    "id_str": "1078969957327798272",
    "indices": [240, 263],
    "media_url": "http://pbs.twimg.com/media/DvlFdaXXgAAmTaH.jpg",
    "media_url_https": "https://pbs.twimg.com/media/DvlFdaXXgAAmTaH.jpg",
    "url": "https://t.co/woDxrz42fp",
    "display_url": "pic.twitter.com/woDxrz42fp",
    "expanded_url": "https://twitter.com/HappyDaysSpain/status/1078969965800316928/photo/1",
    "type": "photo",
    "sizes": {
     "thumb": {
      "w": 150,
      "h": 150,
      "resize": "crop"
     },
     "medium": {
      "w": 842,
      "h": 1024,
      "resize": "fit"
     },
     "large": {
      "w": 842,
      "h": 1024,
      "resize": "fit"
     }, "small": {
      "w": 559,
      "h": 680,
      "resize": "fit"
```

```
        }
       }
      }
     ]
    },
    "extended_entities": {
     "media": [
      {
       "id": 1078969957327798272,
       "id_str": "1078969957327798272",
       "indices": [240, 263],
       "media_url": "http://pbs.twimg.com/media/DvlFdaXXgAAmTaH.jpg",
       "media_url_https": "https://pbs.twimg.com/media/DvlFdaXXgAAmTaH.jpg",
       "url": "https://t.co/woDxrz42fp",
       "display_url":
       "pic.twitter.com/woDxrz42fp",
       "expanded_url":
"https://twitter.com/HappyDaysSpain/status/1078969965800316928/photo/1",
       "type": "photo",
       "sizes": {
        "thumb": {
         "w": 150,
         "h": 150,
         "resize": "crop"
        },
        "medium": {
         "w": 842,
         "h": 1024,
         "resize": "fit"
        },
        "large": {
         "w": 842,
         "h": 1024,
         "resize": "fit"
        },
        "small": {
         "w": 559,
         "h": 680,
         "resize": "fit"
        }
       }
      }
     ]
    },
    "metadata": {
     "iso_language_code": "es",
     "result_type": "recent"
    },
    "source": "<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter for
iPhone</a>",
    "in_reply_to_status_id": null,
    "in_reply_to_status_id_str": null,
```

```
  "in_reply_to_user_id": null,
  "in_reply_to_user_id_str": null,
  "in_reply_to_screen_name": null,
  "user": {
    "id": 930577681,
    "id_str": "930577681",
    "name": "Jose Luis Infantes",
    "screen_name": "HappyDaysSpain",
    "location": "Marbella, Espa\u00f1a",
    "description": "Marketing para Restaurantes: \"Una peque\u00f1a acci\u00f3n da,
siempre, los mejores resultados.\"",
    "url": "https://t.co/nyePW1veYL",
    "entities": {
      "url": {
        "urls": [
          {
            "url": "https://t.co/nyePW1veYL",
            "expanded_url": "https://joseluisinfantes.com",
            "display_url": "joseluisinfantes.com",
            "indices": [0, 23]
          }
        ]
      },
      "description": {
        "urls": []
      }
    },
    "protected": false,
    "followers_count": 8072,
    "friends_count": 705,
    "listed_count": 314,
    "created_at": "Tue Nov 06 21:30:15 +0000 2012",
    "favourites_count": 13728,
    "utc_offset": null,
    "time_zone": null,
    "geo_enabled": true,
    "verified": false,
    "statuses_count": 19461,
    "lang": "es",
    "contributors_enabled": false,
    "is_translator": false,
    "is_translation_enabled": false,
    "profile_background_color": "0099B9",
    "profile_background_image_url": "http://abs.twimg.com/images/themes/theme4/bg.gif",
    "profile_background_image_url_https":
"https://abs.twimg.com/images/themes/theme4/bg.gif",
    "profile_background_tile": false,
    "profile_image_url":
"http://pbs.twimg.com/profile_images/1063751577910034432/42xWcQjs_normal.jpg",
    "profile_image_url_https":
"https://pbs.twimg.com/profile_images/1063751577910034432/42xWcQjs_normal.jpg",
    "profile_banner_url": "https://pbs.twimg.com/profile_banners/930577681/1526119911",
```

```
    "profile_link_color": "0099B9",
    "profile_sidebar_border_color": "5ED4DC",
    "profile_sidebar_fill_color": "95E8EC",
    "profile_text_color": "3C3940",
    "profile_use_background_image": true,
    "has_extended_profile": true,
    "default_profile": false,
    "default_profile_image": false,
    "following": false,
    "follow_request_sent": false,
    "notifications": false,
    "translator_type": "none"
  },
  "geo": null, "coordinates": null,
  "place": {
    "id": "db3ae73fb88e93d8",
    "url": "https://api.twitter.com/1.1/geo/id/db3ae73fb88e93d8.json",
    "place_type": "city",
    "name": "Marbella",
    "full_name": "Marbella, Spain",
    "country_code": "ES",
    "country": "Spain",
    "contained_within": [],
    "bounding_box": {
      "type": "Polygon",
      "coordinates": [
        [
          [-5.0149567, 36.4598413],
          [-4.7308677, 36.4598413],
          [-4.7308677, 36.5551485],
          [-5.0149567, 36.5551485]
        ]
      ]
    },
    "attributes": {}
  },
  "contributors": null,
  "is_quote_status": false,
  "retweet_count": 0,
  "favorite_count": 0,
  "favorited": false,
  "retweeted": false,
  "possibly_sensitive": false,
  "lang": "es"
}
```

B Manual del programador

La URL de GitHub de donde se puede obtener el código desarrollado para LECONORE es: https://github.com/danisantaella/leconore

C Manual de instalación y despliegue

En primer lugar, es necesario tener la imagen que se quiere instalar y el software Docker instalado en el sistema donde se quiere desplegar esta imagen.

Se procede abriendo una terminal, desde la que navegamos al directorio donde se encuentra la imagen. Una vez localizado, se ejecuta el comando:

docker load -i "nombre de la imagen"

Docker se encarga de descargar los módulos necesarios en el sistema. Con el comando ***docker ps*** se comprueba qué contenedores están cargados y ejecutándose.

Para desplegar la imagen en un contenedor, ejecutamos:

docker run -dti --name "nombre del contendor" *-p* 8000:8000 "nombre de la imagen"

donde 8000:8000 es la conexión de puertos, el 8000 de dentro del a aplicación y 8000 del sistema. Por tanto, si todo ha ido bien, en el navegador deberíamos ver una imagen similar a esta por el puerto 8080.

El contenedor se ejecutar automáticamente, entrando con el usuario creado y en su carpeta. Mediante la ejecución del script de inicio ***sh init.sh*** se iniciarán todos los sistemas necesarios y estará́ operativa la aplicación en la dirección de red asignada por el sistema huésped.